May 14, 2013

Randomness in Algorithm Design

Shuji Kijima

Dept. Informatics,
Grad. School of Info. Sci. and Elect. Eng.
Kyushu Univ., Fukuoka Japan

My research field – algorithm design

Randomized Algorithms

- Perfect Sampling (based on MCMC) (w/ Matsui)
- Deterministic random walk
  (w/ Makino, Koga, Kajino, Shiraga, Yamauchi, Yamashita)
- Finding frequent items in a data stream
  (w/ Ogata, Yamauchi, Yamashita)
- Randomized rounding on a permutahedron
  (w/ Hatano, Takimoto, Yaustake, Suehiro, Takeda, Nagano)
- Median stable matching (w/ Nemoto)

Graph Algorithms

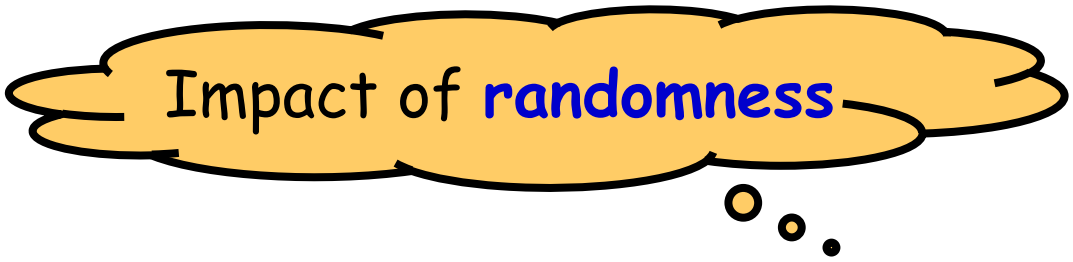
- Is subgraph isomorphism in an interval graph P? NP-c?
  (w/ Saitoh, Otachi, Uno)
- Is counting dominating sets in an interval graph P? #P-c?
  (w/ Okamoto, Uno)
- Can any rigidity circuit with maximum degree 4 be decomposed into two edge-disjoint Hamiltonian paths? (w/ Tanigawa)

Question

"What is the property that

a randomized algorithm *really* requires for randomness?"

1. Impact of randomness: ex. "finding frequent items in a stream"
2. Advanced prob. algo.: "Markov chain Monte Carlo" for sampling comb. obj.
3. Derandomization: "deterministic random walk"



[ISAAC 2011]

1. Finding frequent items in stream

Masatora Ogata, Yukiko Yamauchi,

Shuji Kijima, Masafumi Yamshita

Kyushu Univ.

θ: param. of "freq."

Σ: finite set of items

Finding frequent items

Prob. Finding frequent items

Prescribed: $\theta \in (0,1)$, Input: $\mathbf{x} = (x_1,\ldots,x_N) \in \Sigma^N$ (sequentially)

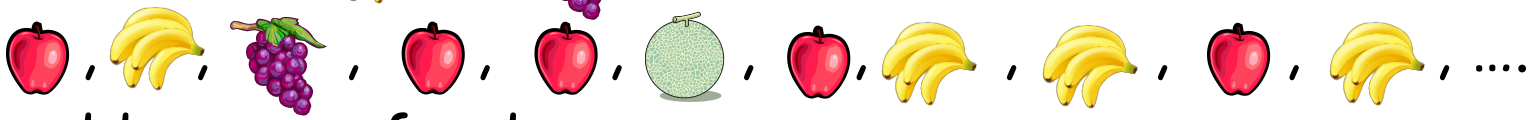
Find: all $s \in \Sigma$ s.t. $f(s) \geq \theta \cdot N$

where f(s) denotes #of s in **x**.

Remark
We do not know
N (or approx. log N)
in advance.

ex1. POS data (@fruit shop) of a day

Σ ={  }

x = 

ex2. IP addresses of a day

$\Sigma \subseteq$ {0.0.0.0,…,255.255.255.255}

x = 123.45.67.89, 111.11.1.1., 123.45,67,89, 122.122.12.12…

would like to find items appearing w/ frequency more than θ=1% of N.

The space complexity of "finding frequent items"

Thm [Karp, Shenker, Papadimitriou '03]

Any online algorithm requires

$\Omega(|\Sigma| \log (N/|\Sigma|))$ bits for exact solution.

(where $N \gg |\Sigma| \gg 1/\theta$)

lower bound for exact algo.

Thm. [Karp, Shenker, Papadimitriou '03]

There is a false-positive algorithm using

$O((1/\theta) \log N)$ bits.

(where $N \gg |\Sigma| \gg 1/\theta$)

deterministic

$o(\log N)$ bits algorithm?

- e.g. $O(\log \log N)$ bits?

Simplified issue: count elem. using o(log N) bits?

Prob: counting elements

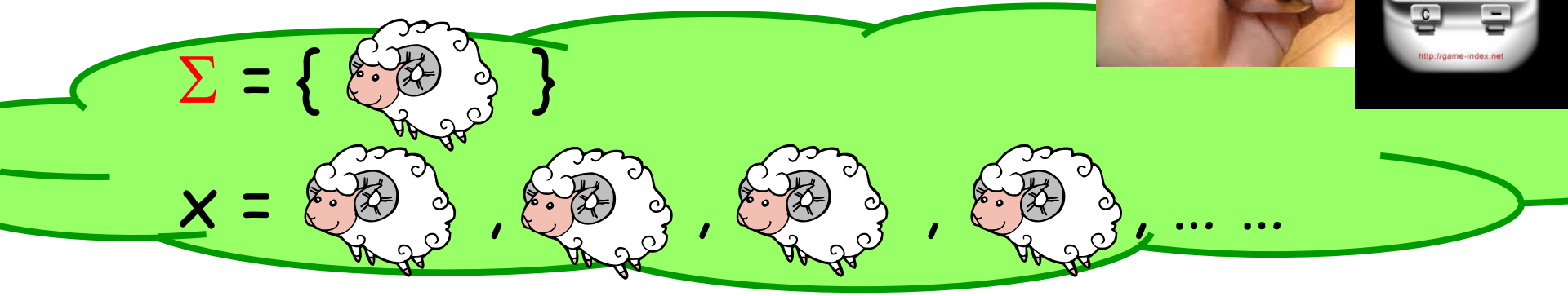
Input: $\mathbf{x} = (a,\ldots,a) \in \Sigma^N$ (sequentially)

Find: N

Remark
we do not know
N (or, approx. log N)
in advance.

Algorithm: exact counting

0. **Set** n:= 0.
1. **Read** an input. **If** no more input, **goto** 3.
2. n++, **Goto** 1.
3. **Output** n (as N = n).



Simplified issue: count elem. using o(log N) bits?

Prob: counting elements

Input: $\mathbf{x} = (a,\ldots,a) \in \Sigma^N$ (sequentially)

Find: N

Remark
we do not know N (or, approx. log N) in advance.

Algorithm: exact counting

0. **Set** n:= 0.
1. **Read** an input. **If** no more input, **goto** 3.
2. n++, **Goto** 1.
3. **Output** n (as N = n).

$\Theta(\log N)$ bits

o(log N) bits approx. algorithm ?

- e.g. O(log log N) bits?

Simplified issue: count elem. using o(log N) bits?

Prob: counting elements

Input: $\mathbf{x} = (a,\ldots,a) \in \Sigma^N$ (sequentially)

Find: N

Remark
we do not know
N (or, approx. log N)
in advance.

Algorithm: exact counting

0. **Set** n:= 0.
1. **Read** an input. **If** no more input, **goto** 3.
2. n++, **Goto** 1.
3. **Output** n (as N = n).

Θ(log N) bits

Thm. (cf. [Flajolet '85, Ogata et al. '11])
Impossible by a *deterministic* algorithm
using o(log N) bits!

Clue for count. algo using O(loglog N) bits?

Prob: counting elements

Input: $x = (a,\ldots,a) \in \Sigma^N$ (sequentially)

Find: N

Remark
we do not know N (or, approx. log N) in advance.

Remark

- Exact val. of N is represented using O(log N) bits.
  - ✓ N = 1,351,127,649,213
- Approx. val. of N is represented using O(log log N) bits
  - ✓ $N \approx 1.351 \times 10^{12}$

const. size "**mantissa**" + "**exponent**" using O(loglog N) bits

o(log N) bits approx. algorithm ?

➢ e.g. O(log log N) bits?

Why do we think o(log N) count. algo?

Prob: counting elements

Input: $\mathbf{x} = (a,\ldots,a) \in \Sigma^N$ (sequentially)

Find: N

Remark
we do not know
N (or, approx. log N)
in advance.

Remark

- Exact val. of N is represented using $O(\log N)$ bits.
  - ✓ N = 1,351,127,649,213
- Approx. val. of N is represented using $O(\log \log N)$ bits
  - ✓ $N \approx 1.351 \times 10^{12}$

So, our question for simplified issue is,

"Can we compute exponent (=log N) using o(log N) bits?"

We have a representation with $O(\log \log N)$ bits.

Simplified issue: count elem. using o(log N) bits?

Prob: counting elements

Input: $\mathbf{x} = (a,\ldots,a) \in \Sigma^N$ (sequentially)

Find: N

Remark
we do not know
N (or, approx. log N)
in advance.

Algorithm: exact counting

0. **Set** n:= 0.
1. **Read** an input. **If** no more input, **goto** 3.
2. n++, **Goto** 1.
3. **Output** n (as N = n).

$\Theta(\log N)$ bits

Thm. (cf. [Flajolet '85, Ogata et al. '11])
Impossible by a *deterministic* algorithm
using o(log N) bits even for approx. count!

[Morris '78]

Probabilistic counting using O(log log N) bits.

Prob: counting elements

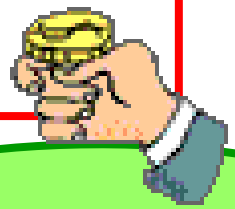
Input: $x = (a,\ldots,a) \in \Sigma^N$ (sequentially)

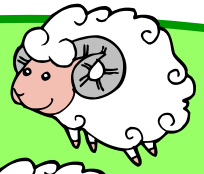
Find: N

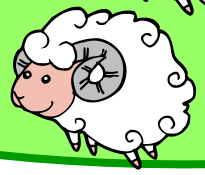
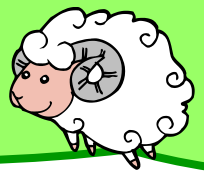
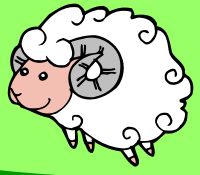
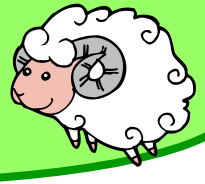
Remark
we do not know N (or, approx. log N) in advance.

Algorithm: *probabilistic* counting

0. **Set** k:= 0.
1. **Read** an input. **If** no more input, **goto** 3.
2. k++, w.p. $1/2^k$. **Goto** 1.
3. **Output** k (as $N \approx 2^k$).



Σ = {  }

X =  ,  ,  ,  , ...

[Morris '78]

Probabilistic counting using O(log log N) bits.

Prob: counting elements

Input: $\mathbf{x} = (a,\ldots,a) \in \Sigma^N$ (sequentially)

Find: N

Remark
we do not know
N (or, approx. log N)
in advance

Algorithm: *probabilistic* counting

0. **Set** k:= 0.
1. **Read** an input. **If** no more input, **goto** 3.
2. k++, w.p. $1/2^k$. **Goto** 1.
3. **Output** k (as $N \approx 2^k$).

⇒ key point
"w.p. $1/2^k$ " using
O(log K) bits on PTM.

O(log log N) bits

Thm. [Morris '78, Flajolet '85]
$E[2^{k+1}] \approx N+1$

because
$N \approx 1+2+4+8+16+\ldots+2^k$

[Ogata et al '11]

Improved prob. counting using O(log log N) bits.

Prob: counting elements

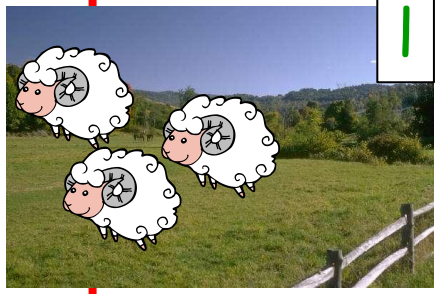
Input: $\mathbf{x} = (a,\ldots,a) \in \Sigma^N$ (sequentially)

Find: N

Remark
we do not know
N (or, approx. log N)
in advance.

Algorithm: *improved* probabilistic counting

0. **Set** k:= 0, l=0.
1. **Read** an input. **If** no more input, **goto** 4.
2. l++, w.p. $1/2^k$.
3. If l==C, k++, and **set** l := l' w.p. $\binom{C}{l'} \cdot \frac{1}{2^C}$. **Goto** 1.
4. **Output** l and k (as $N \approx l*2^k$).



Thm. [Ogata, Yamauchi, K., Yamashita '11]

$E[l*2^k] \approx N.\ Pr[|l*2^k - N|/N < \varepsilon] > 1-\delta$

[Ogata et al '11]

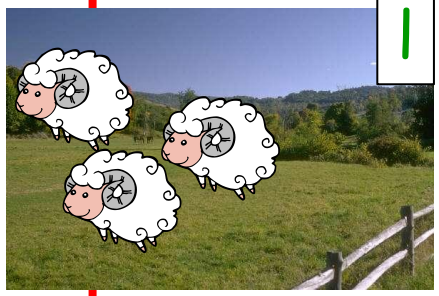
Improved prob. counting using O(log log N) bits.

Prob:

Find:

suppose we stores "l" (< 2^b) as an undecimal,

i.e., "l" is a pool storing sheep (w.p.$1/2^k$).



Algorithm: *improved* probabilistic counting

0. **Set** k:= 0, l=0.
1. **Read** an input. **If** no more input, **goto** 4.
2. l++, w.p. $1/2^k$.
3. If l==C, k++, and **set** l := l' w.p. $\binom{C}{l'} \cdot \frac{1}{2^C}$. **Goto** 1.
4. **Output** (est. N ~ l*2^k)

this algorithm stores each sheep (in "l") w.p. $1/2^k$ at any k;
because a sheep stored when "exponent"=j w.p. $1/2^j$,
is still stored when "exponent"=k (k > j), w.p. $1/2^{k-j}$
$\Rightarrow$ Pr[sheep in l] = $1/2^j * 1/2^{k-j} = 1/2^k$.

[Ogata et al '11]

Improved prob. counting using O(log log N) bits.

Prob: counting elements

Input: $\mathbf{x} = (a,\ldots,a) \in \Sigma^N$ (sequentially)

Find: N

Remark
we do not know
N (or, approx. log N)
in advance.

Algorithm: *improved* probabilistic counting

0. **Set** k:= 0, l=0.
1. **Read** an input. **If** no more input, **goto** 4.
2. l++, w.p. $1/2^k$.
3. If l==C, k++, and **set** l := l' w.p. $\binom{C}{l'} \cdot \frac{1}{2^C}$. **Goto** 1.
4. **Output** l and k (as $N \approx l*2^k$).

O(log log N) bits

Thm. [Ogata, Yamauchi, K., Yamashita '11]

$E[l*2^k] \approx N$. $Pr[|l*2^k - N| \ge$

finding frequent items in a similar way.

Finding frequent items

Prob. Finding frequent items

Prescribed: $\theta \in (0,1)$, Input: $\boldsymbol{x} = (x_1,\ldots,x_N) \in \Sigma^N$ (sequentially)

Find: all $s \in \Sigma$ s.t. $f(s) \geq \theta \cdot N$

where $f(s)$ denotes #of s in $\boldsymbol{x}$.

Remark
We do not know
N (or approx. log N)
in advance.

Thm. [Ogata, Yamauchi, K., Yamashita '11]

There exists a randomized algo. using $O(\log \log N)$ bits.

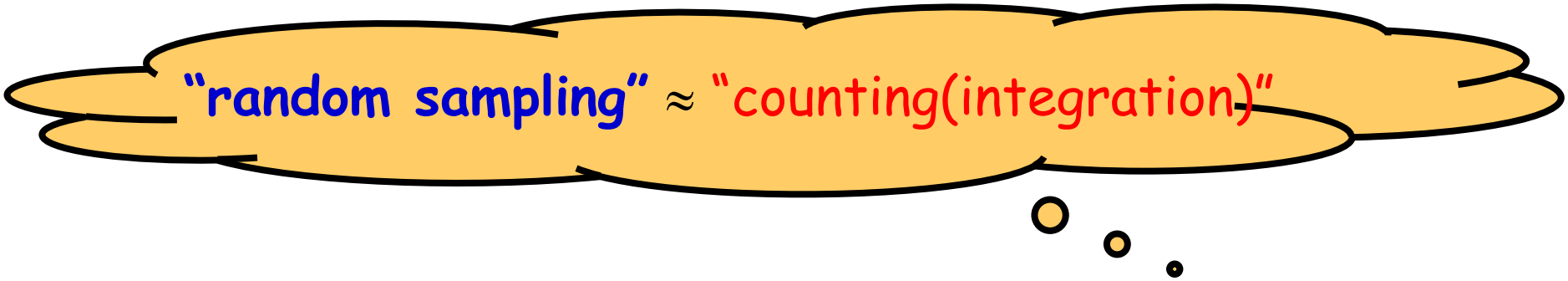
for finding frequent items.

omit the detail

$o(\log N)$ bits algorithm?

- e.g. $O(\log \log N)$ bits?

Advanced technique



2. Random Sampling of Combinatorial Objects

Markov chain Monte Carlo (MCMC)

joint with **Tomomi Matsui** (Tokyo Inst. Tech.)

Ex. 1. Contingency table

- ✓ matrix of non-negative integers
- ✓ satisfies (given) marginal sums

						12
						18
5	4	3	7	5	6	30

5	4	3	0	0	0	12
0	0	0	7	5	6	18
5	4	3	7	5	6	30

table A

4	3	1	3	1	0	12
1	1	2	4	4	6	18
5	4	3	7	5	6	30

table B

0	0	0	1	5	6	12
5	4	3	6	0	0	18
5	4	3	7	5	6	30

table C

Problem

Given: marginal sums

Output: a contingency table u.a.r.

Ex. 1. Contingency table

- ✓ matrix of non-negative integers
- ✓ satisfies (given) marginal sums

						12
						18
5	4	3	7	5	6	30

counting # of tables satisfying given marginal sums

$\Rightarrow$ #P-complete (NP-hard), even when size of table is $2 \times n$

[Dyer, Kannan, and Mount '97]

Problem

Given: marginal sums

Output: a contingency table u.a.r.

sampling via Markov chain

Ex. 1. Markov chain for contingency tables [KM '06]

transition rule is defined as follows

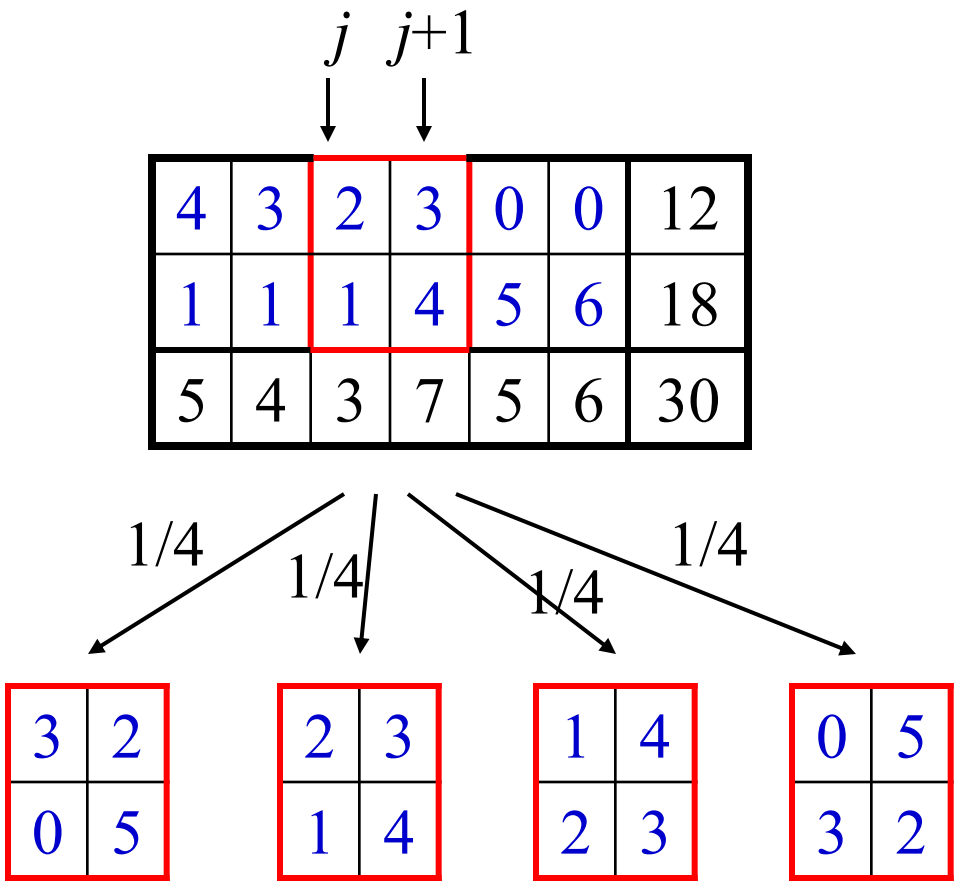
1. choose a consecutive pair of columns $(j, j+1)$ u.a.r. (prob. $1/(n-1)$)
2. change the values of cells in $(j, j+1)$-th columns u.a.r. on possible states

2	3	5
1	4	5
3	7	10

+

$+k$	$-k$
$-k$	$+k$

=> preserve marginal sums



4 possible states

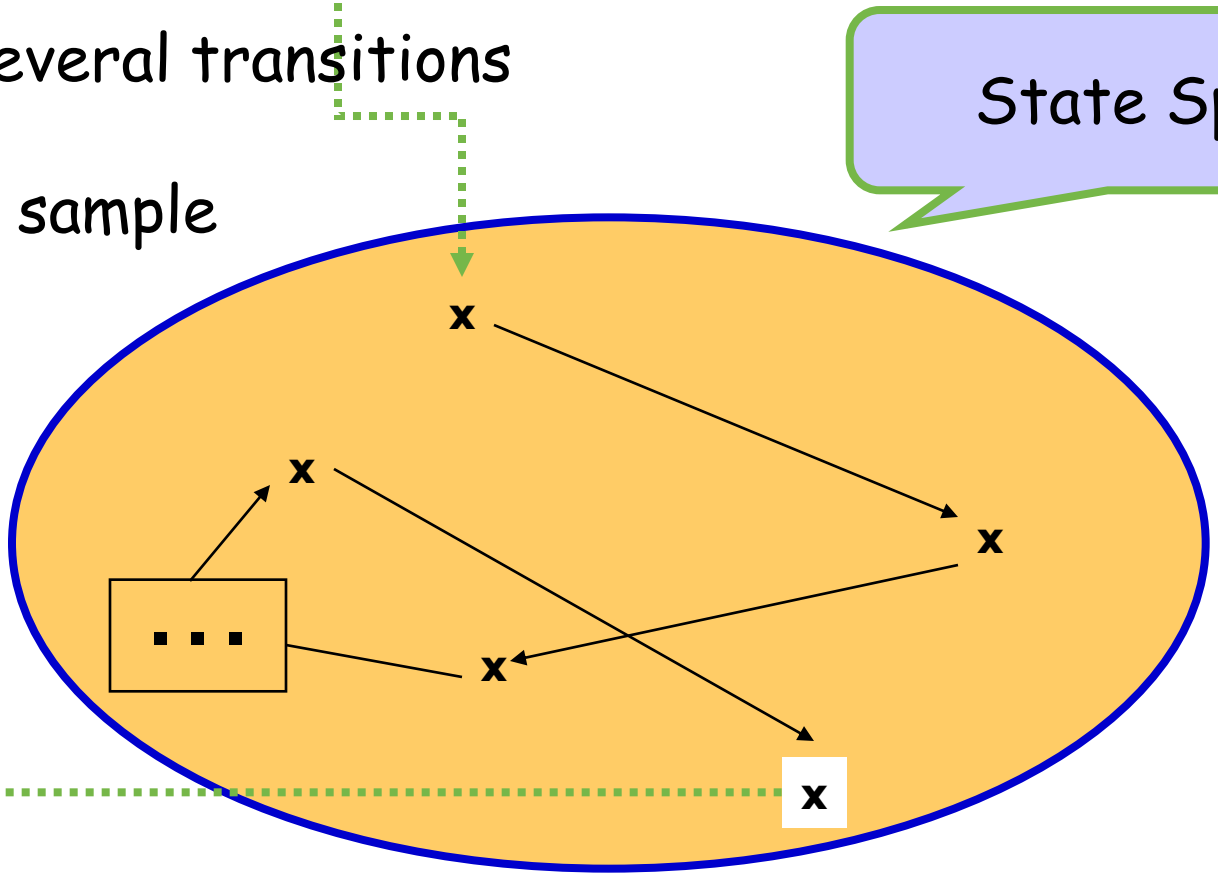
(requirement on non-negativity)

Basic idea of "sampling via Markov chain"

Start from arbitrary initial state

Make several transitions

Output a sample



The output is 'approximately' according to the stat. dist.

Our Markov chain for contingency tables [KM '06]

Them

Our Markov chain is ergodic, and

the unique stat. dist. of the chain is uniform dist.

proof for the latter claim: detailed balance equations

$$1 \cdot P(x,y) = 1 \cdot P(y,x) \qquad \forall x, y \in \Omega$$ hold.

with the following ex. of a pair (x,y)

X

4	3	2	3	0	0	12
1	1	1	4	5	6	18
5	4	3	7	5	6	30

$\frac{1}{5} \cdot \frac{1}{4}$

$\frac{1}{5} \cdot \frac{1}{4}$

Y

4	3	0	5	0	0	12
1	1	3	2	5	6	18
5	4	3	7	5	6	30

transition prob. from X to Y

transition prob. from Y to X

since ...

Our Markov chain for contingency tables [KM '06]

Them

Our Markov chain is ergodic, and

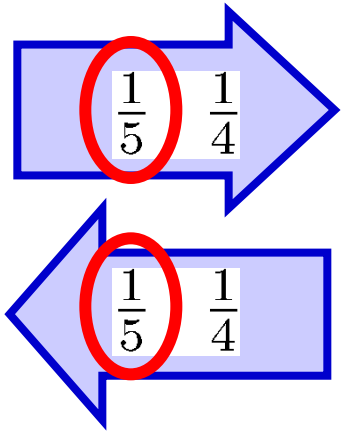
the unique stat. dist. of the chain is uniform dist.

proof for the latter claim: detailed balance equations

$$1 \cdot P(x,y) = 1 \cdot P(y,x) \qquad \forall x, y \in \Omega$$ hold.

X

4	3	2	3	0	0	12
1	1	1	4	5	6	18
5	4	3	7	5	6	30



Y

4	3	0	5	0	0	12
1	1	3	2	5	6	18
5	4	3	7	5	6	30

choose a consecutive pair of indices u.a.r. (w.p. 1/(6-1))

Our Markov chain for contingency tables [KM '06]

Them

Our Markov chain is ergodic, and

the unique stat. dist. of the chain is uniform dist.

proof for the latter claim: detailed balance equations

$$1 \cdot P(x,y) = 1 \cdot P(y,x) \qquad \forall x, y \in \Omega$$ hold.

X

4	3	2	3	0	0	12
1	1	1	4	5	6	18
5	4	3	7	5	6	30

$\frac{1}{5}$ $\frac{1}{4}$

$\frac{1}{5}$ $\frac{1}{4}$

Y

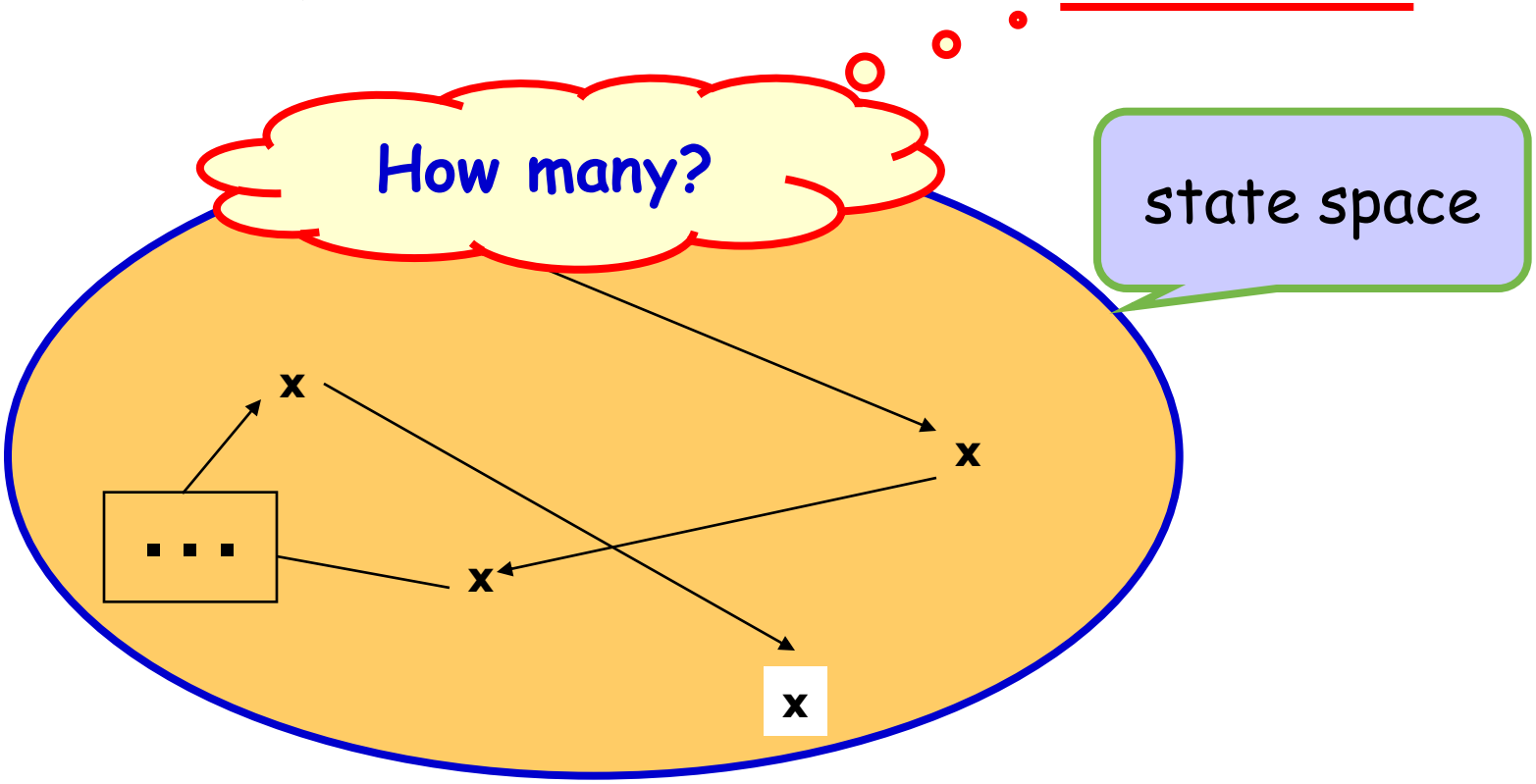
4	3	0	5	0	0	12
1	1	3	2	5	6	18
5	4	3	7	5	6	30

on the condition (3,4) columns are chosen,
there are common 4 possible states,
since values in other columns are same.

3	2	2	3	1	4	0	5
0	5	1	4	2	3	3	2

Basic idea of "sampling via Markov chain"

1. Design a Markov chain whose stat. dist = aiming dist.
2. Generate a sample from stat. dist. after many tran.s.



outputs after many transitions according to asymptotically stationary distribution

"How many transitions do we need?"

- For **approximate sampling**,
  - estimate mixing time, and
    bound the discrepancy (by total variation distance)

- For **perfect sampling** in finite time!?
  - meaning "sampling exactly according to the stat. distr."

  $\Rightarrow$ "Coupling from The Past (CFTP)" [Propp & Wilson 96]
  is an ingenious simulation of MC, and
  realizes perfect sampling!

CFTP Algorithm and Theorem. [Propp & Wilson 96]

Markov chain *MC*:
- Ω: finite state space
- $\Phi_s^t(x, \lambda)$: transition rule
- ergodic

CFTP Algorithm

1. Set $T = -1$; set λ: empty;
2. Generate $\lambda[T]$,... , $\lambda[T/2 - 1]$: random number;
   Put $\lambda := (\lambda[T],... , \lambda[T/2 - 1], \lambda[T/2],\ldots,\lambda[-1])$;
3. Start a chain from every element in Ω at period T, run *MC* with λ to period 0.
   a. if coalesce ($\exists y \in \Omega, \forall x \in \Omega, y = \Phi_T^0(x, \lambda)) \Rightarrow$ return y;
   b. otherwise, set $T := T$-1; go to step 2.;

CFTP Theorem

When CFTP Algorithm terminates, the returned value realizes the random variable from stationary distribution, exactly.

what is the idea of CFTP?

Idea of CFTP (Coupling From the Past)

- Suppose an ergodic chain from infinite past, imaginarily.
  - Present state (state at time 0) is

    EXACTLY according to the stat. dist.
- What is the present state?
  - Guess from the recent random transitions.

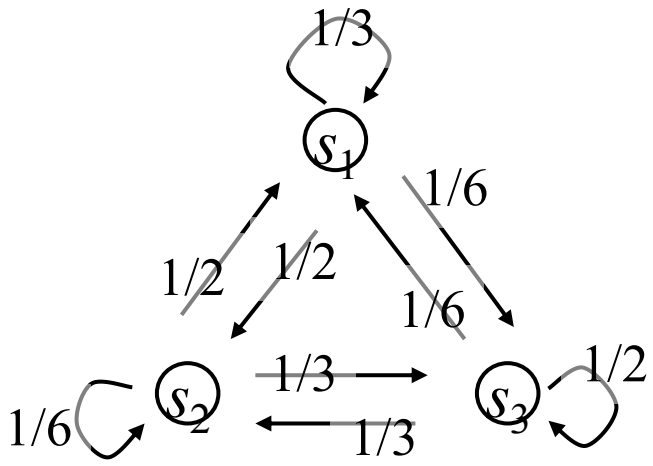
    $\Rightarrow$ Find the evidence of the present state.

obtained by considering random numbers and transitions with an update function.

Update function -- with an example

- An ergodic Markov chain *MC*

We consider to determine the next state with
- a random number $\lambda \in \{1,\ldots,6\}$ (u.a.r.), and
- an Update function.

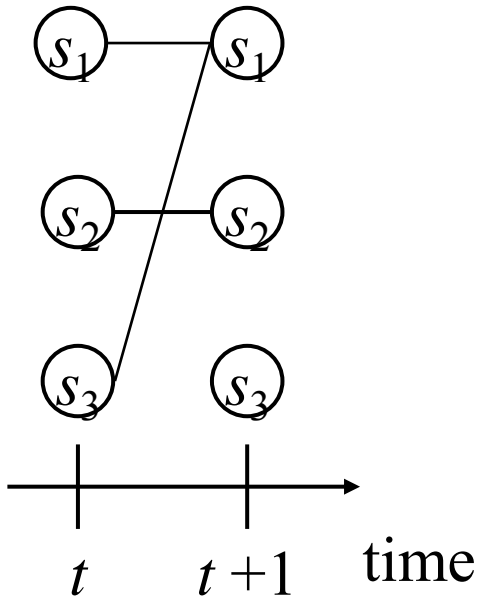


current state

λ	s_1	s_2	s_3
1	s_3	s_1	s_2
2	s_2	s_3	s_2
3	s_2	s_1	s_3
4	s_1	s_1	s_3
5	s_1	s_2	s_1
6	s_2	s_3	s_3

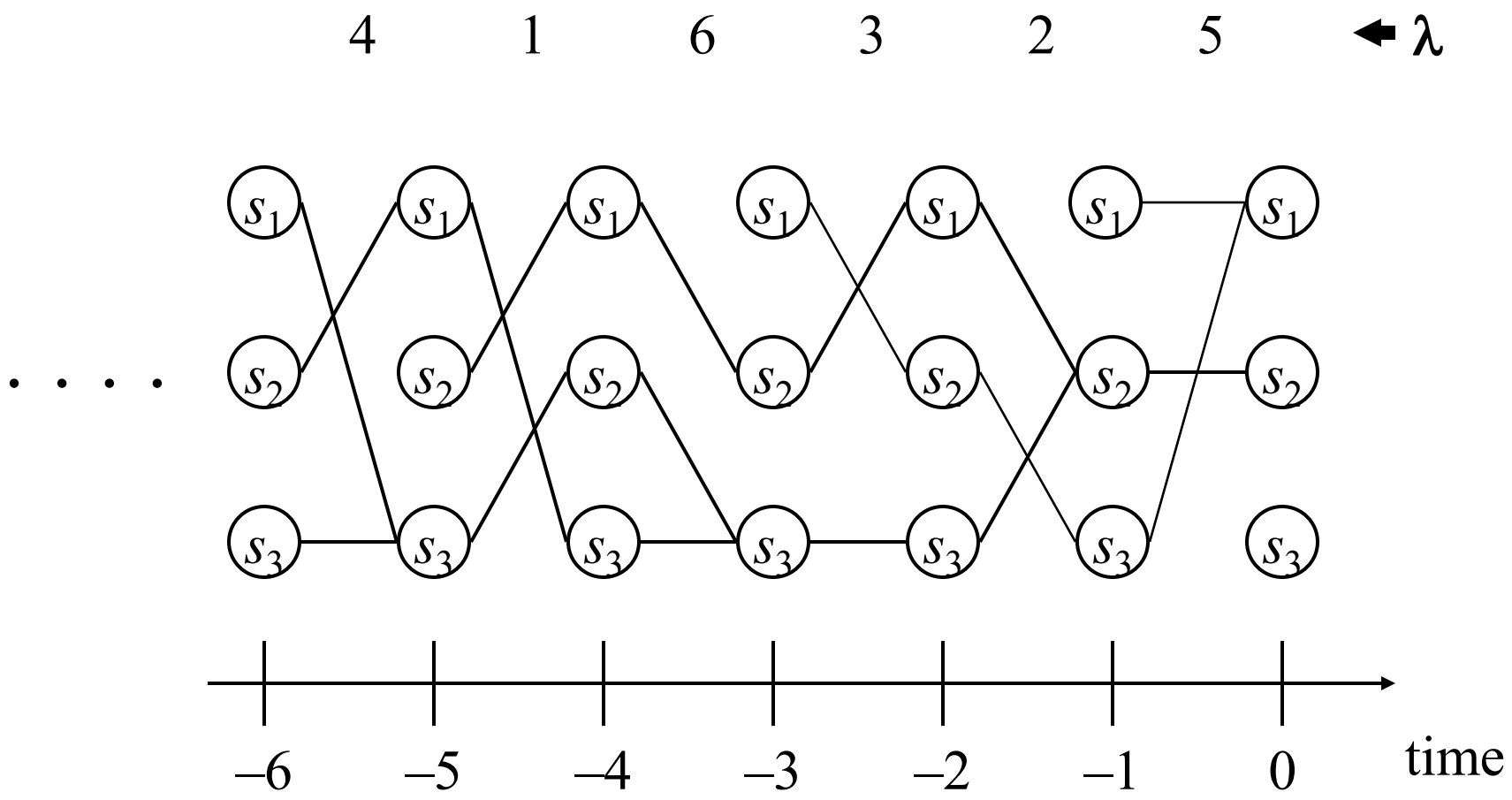
This table shows an update function

e.g., 5 ← λ



This is an illustration of a transition.

Figure of CFTP

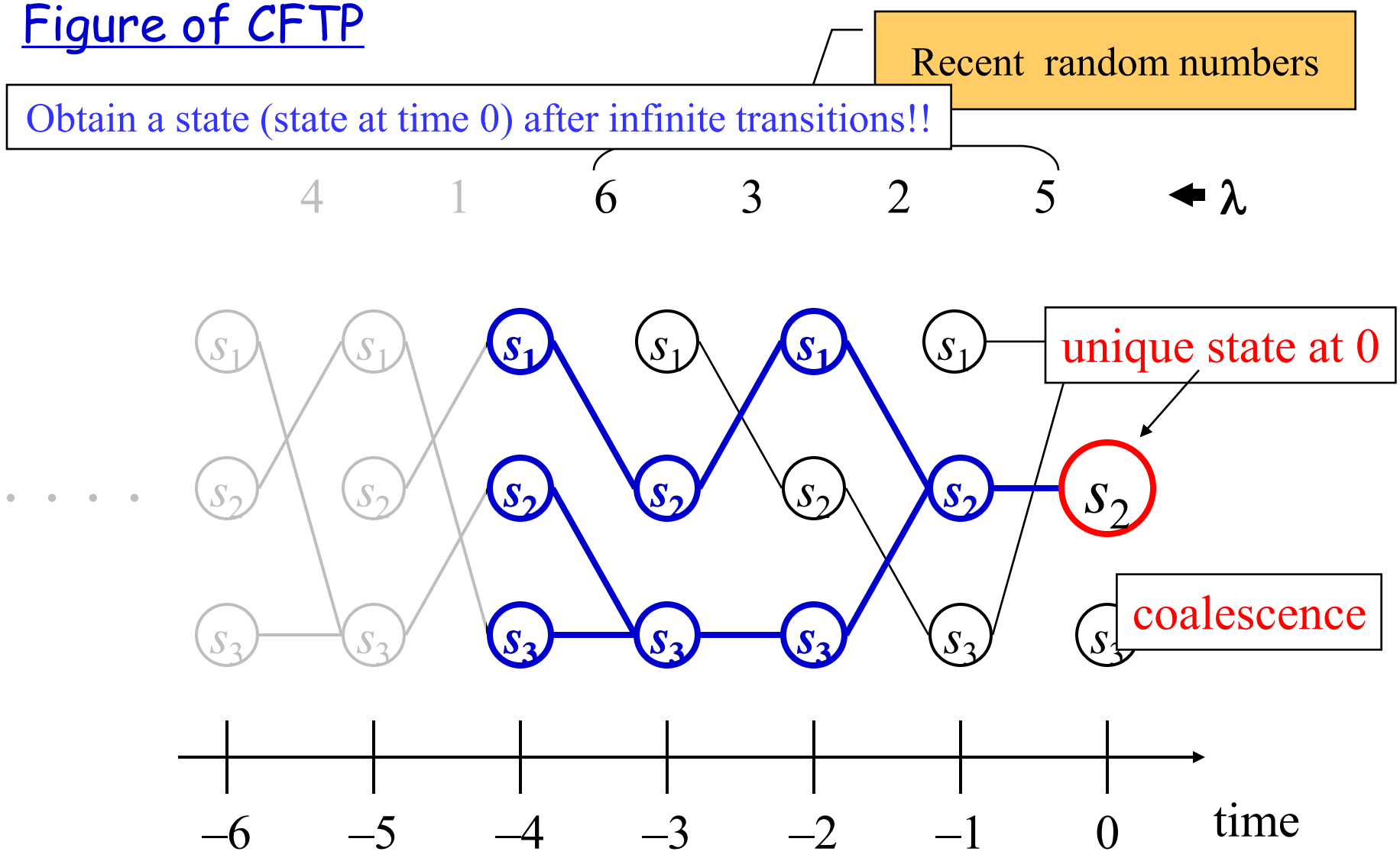


Suppose a Markov chain starting at infinite past

Thus the present state is according to the stationary distribution, exactly.

=> We'd like to know the state at time 0.

Figure of CFTP

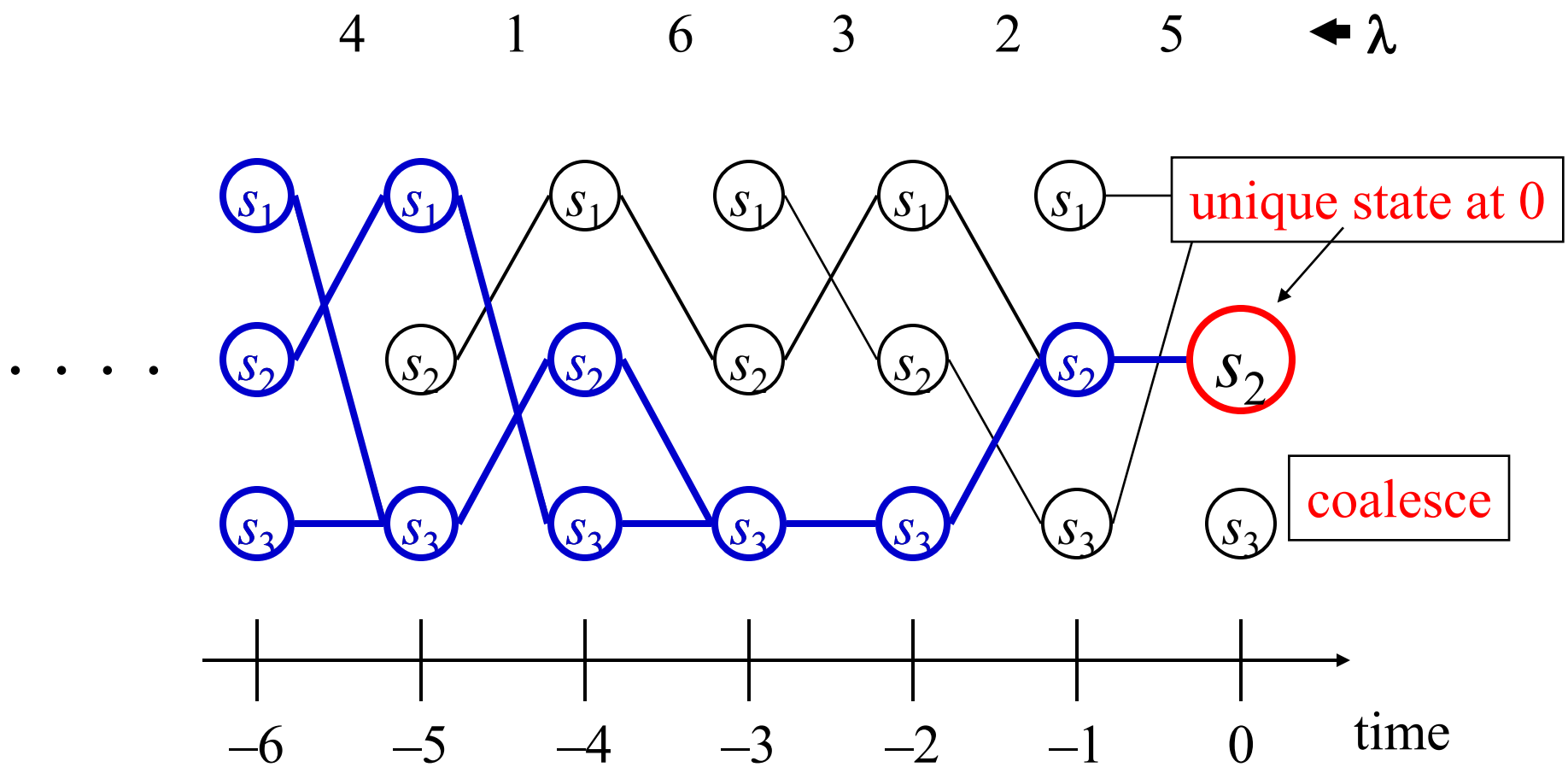


Suppose we know some “recent” transitions.

We may obtain a unique state at time 0. we call this situation ‘coalescence’

Figure of CFTP

States in ancient times (more than coalescence time) don't matter at time 0.



Even if we start the simulation before -4, the result of a simulation must be s_2, since we tested all possibility at time -4.

CFTP Algorithm and Theorem. [Propp & Wilson 96]

Markov chain *MC*:
- Ω: finite state space
- $\Phi_s^t(x, \boldsymbol{\lambda})$: transition rule
- ergodic

CFTP Algorithm

1. Set $T = -1$; set $\boldsymbol{\lambda}$: empty;
2. Generate $\lambda[T]$,... , $\lambda[T/2 - 1]$: random number;
   Put $\boldsymbol{\lambda} := (\lambda[T],... , \lambda[T/2 - 1], \lambda[T/2],\ldots,\lambda[-1])$;
3. Start a chain from every element in Ω at period T, run *MC* with $\boldsymbol{\lambda}$ to period 0.
   a. if coalesce ($\exists y \in \Omega, \forall x \in \Omega, y = \Phi_T^0(x, \boldsymbol{\lambda})$) $\Rightarrow$ return y;
   b. otherwise, set $T := T$-1; go to step 2.;

CFTP Theorem

When CFTP Algorithm terminates, the returned value realizes the random variable from stationary distribution, exactly.

what is the idea of CFTP?

to design an efficient perfect sampler based on CFTP

monotone Markov chain

- ✓ Ising model [Propp Wilson 96]
- ✓ Potts model (clutter model)
- ✓ Tiling [Propp 97]
- ✓ Two-rowed contingency tables [K, T. Mastui: RSA06]
- ✓ Discretized Dirichlet distribution [T. Mastui, K. 07]
- ✓ Queueing Networks (Jackson net, BCMP) [K,T. Mastui SICOMP08]
- ✓ Q-Ising model [Yamamoto, K, Y.Mastui JCO11]

Another CFTP

- ✓ (Rooted) spanning trees [Propp Wilson 98]

Improvement of memory space

- ➢ Read once algorithm [Wilson 00]
- ➢ Interruptible algorithm [Fill 98]

Poly-time Sampling --recent develop. & open prob.

Poly-time algorithms for

- ✓ uniform/log-concave on high-dim. convex body.
  [Dyer, Frieze, & Kannan 91], [Lovasz & Vempala 07]
- ✓ perfect bip. matchings (a.k.a. permanent)
  [Jerrum, Sinclair, & Vigoda 04], [Stefankovic, Vempala, & Vigoda 09]
- ✓Ising model (b/w pic) [Jerrum & Sinclair 93], [Propp &Wilson 96]

Open problems (poly-time algo.):

- ✓ **mxn contingency tables** (u.a.r.)
- ✓ **ideals of a poset** (u.a.r.)
- ✓ **Tutte poly.**
  - ➢ **matroid bases** (u.a.r.)
  - ➢ **Potts model**

[K 09+]

Impossibility

Unless RP=NP,

No poly time sampler for **gen. log-supermodula distr.**

"What is the property that
a randomized algorithm *really* requires for randomness?"

For the goal of derandomizing MCMC (in progress yet)

3. Deterministic Random Walk

Shuji Kijima

Grad. Sch. of Info. Sci. & Elec. Eng.,

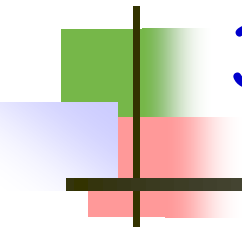
Kyushu Univ.

joint with

Kentaro Koga, Kazuhisa Makino (2010+, ANALCO 2012)

Hiroshi Kajino, Kazuhisa Makino (2012+)

Takeharu Shiraga, Yukiko Yamauchi, Masafumi Yamashita (2012+)



3.1. What does "deterministic RW" mean?

Introduction of the *rotor-router* model

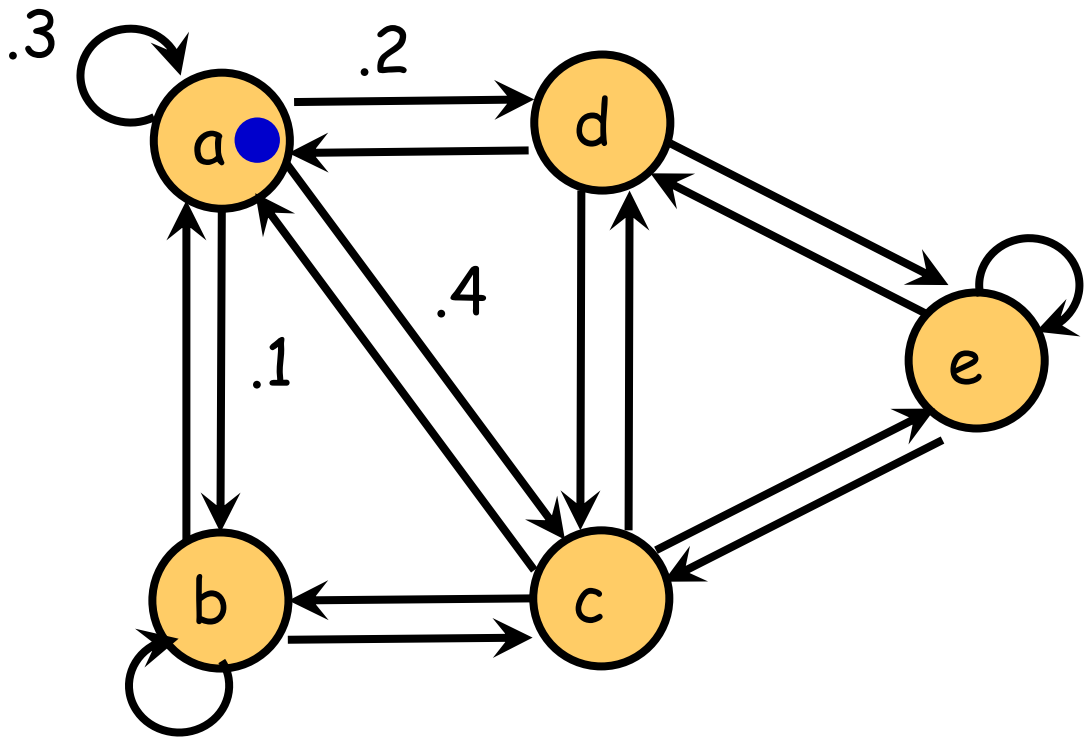
(a.k.a. Propp machine [Propp '00]) on finite graphs

--- from a viewpoint of an analogy of a random walk.

In Random Walk

A token randomly walks on a graph.

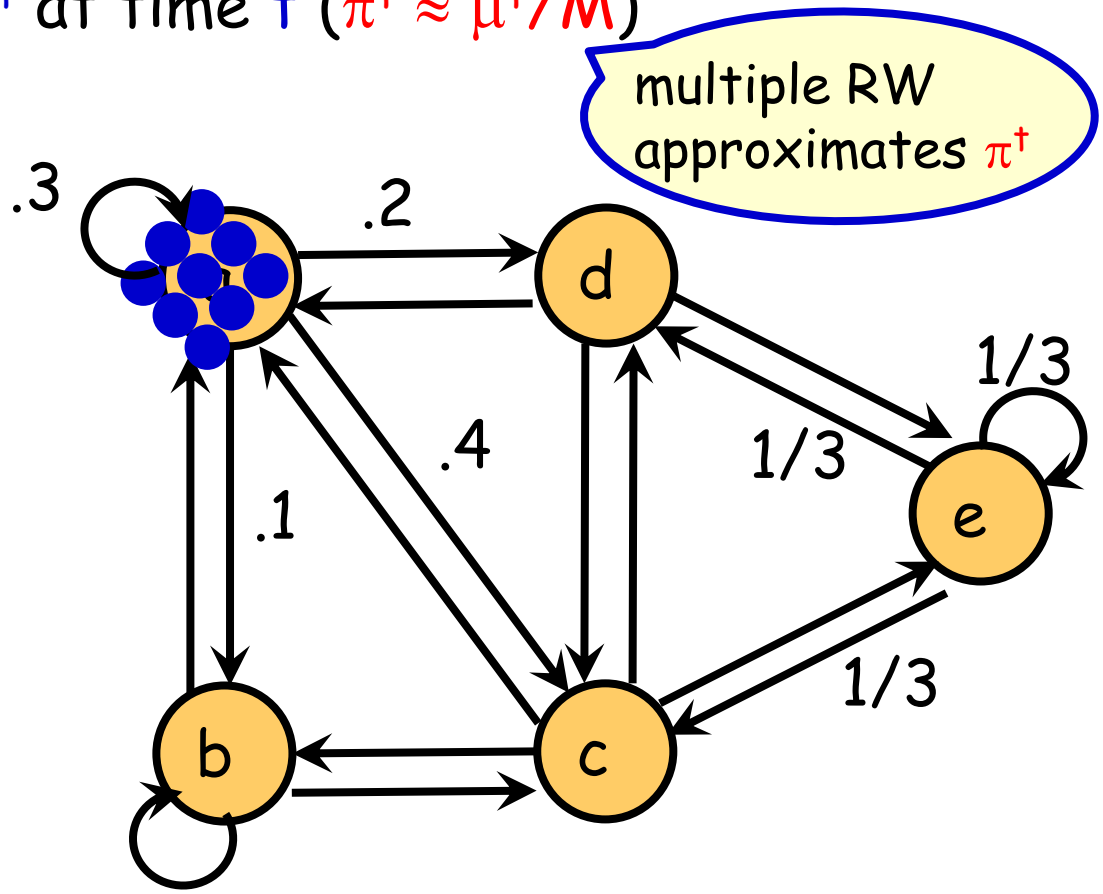
- ✓ π^0: ini. distr. (token is on v at time 0, w.p. $(\pi^0)_v$)
- ✓ P: trans. prob. matrix (move u-> v w.p. P_{uv})
- ✓ distr. $\pi^t := \pi^0 P^t$ at time t (token is on v at time t w.p. $(\pi^t)_v$)



In multiple RW (RW of many tokens)

M tokens randomly walk on a graph, independently.

- ✓ μ^0: ini. config. ($\pi^0 \approx \mu^0/M$)
- ✓ P: trans. prob. matrix (a token moves u->v w.p. P_{uv})
- ✓ exp. config. $\mu^t := \mu^0 P^t$ at time t ($\pi^t \approx \mu^t/M$)



In multiple RW (RW of many tokens)

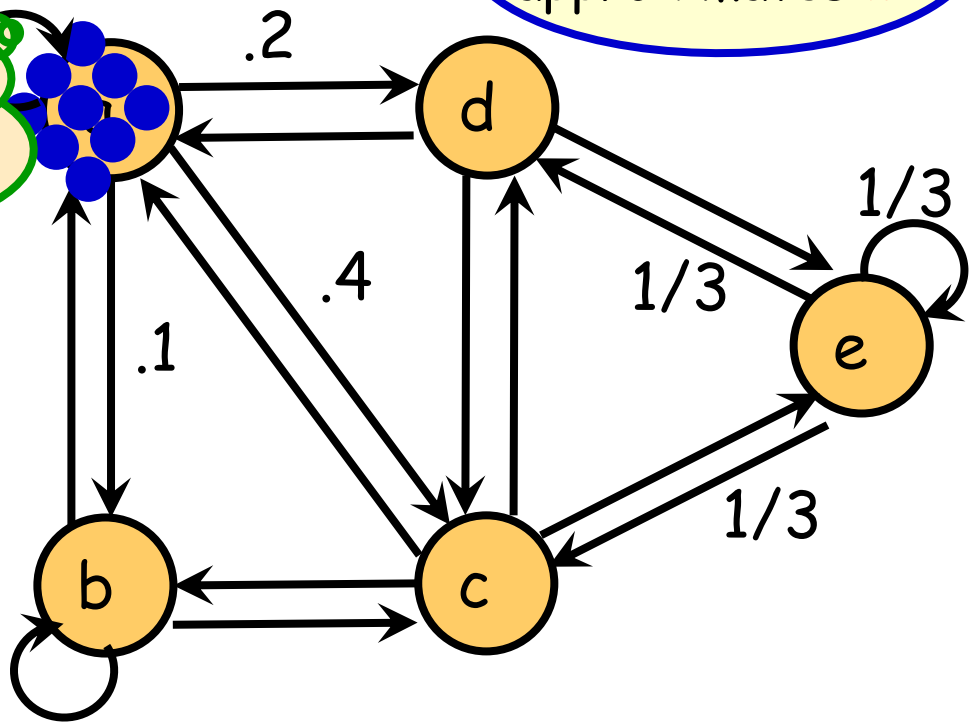
M tokens randomly walk on a graph, independently.

- ✓ μ^0: ini. config. ($\pi^0 \approx \mu^0/M$)
- ✓ P: trans. prob. matrix (a token moves u->v w.p. P_{uv})
- ✓ exp. config. $\mu^t := \mu^0 P^t$ at time t ($\pi^t \approx \mu^t/M$)

multiple RW approximates π^t

(When N is large,)
tokens move to a,b,c,d
with ratios approx. .3,.1,.4,.2

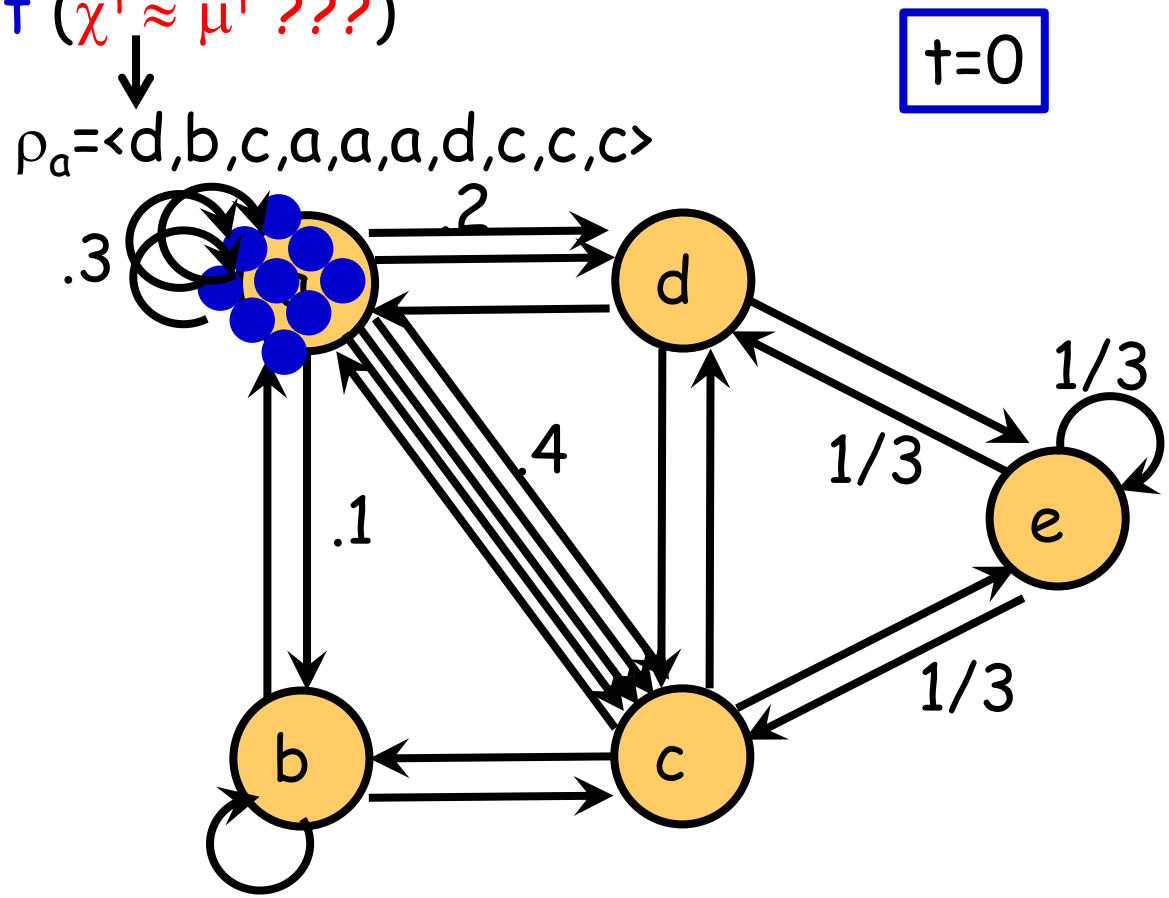
Can we derandomize?



Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

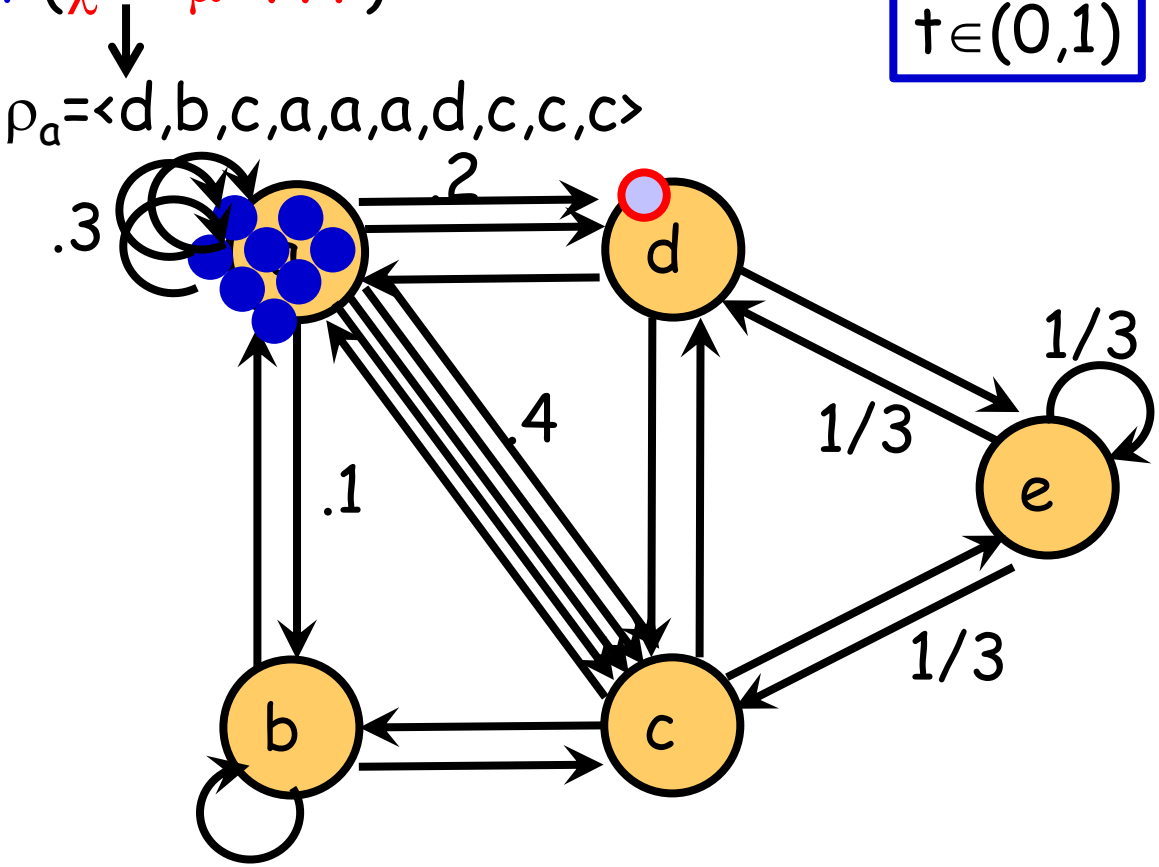


Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

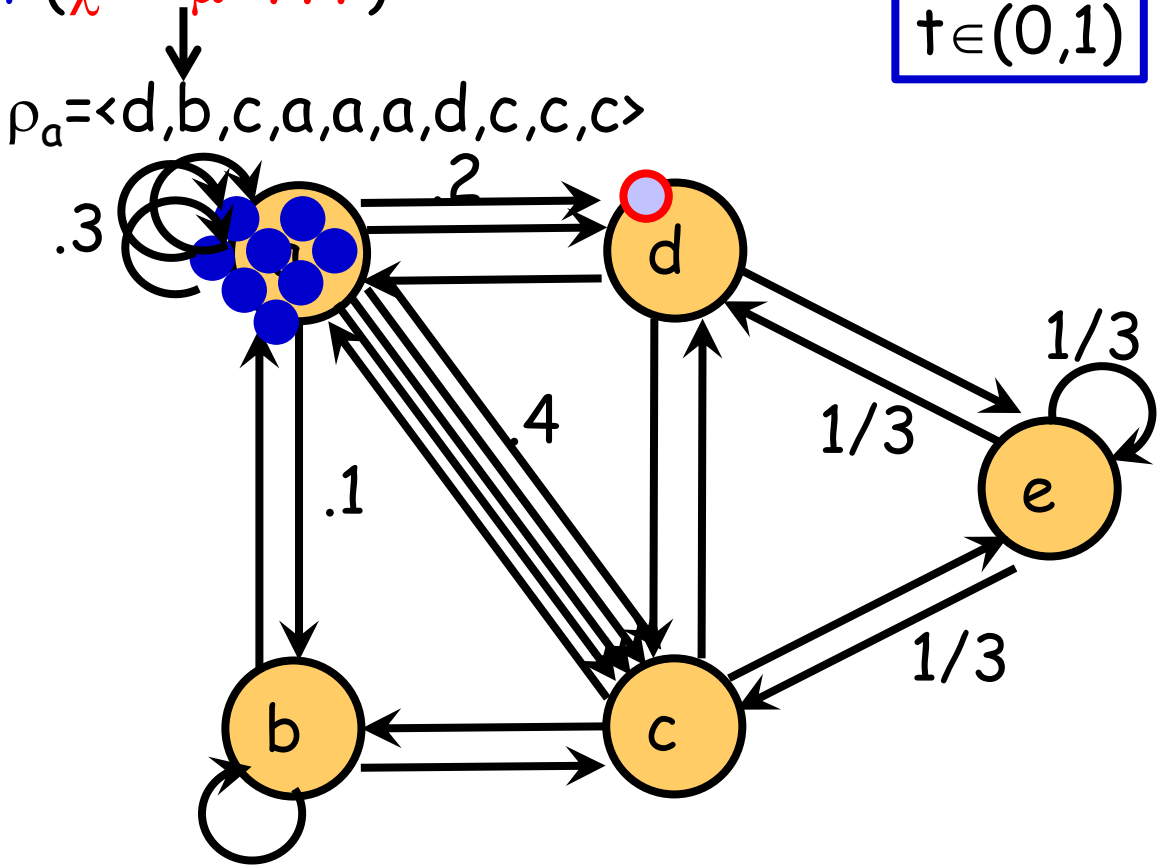


Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

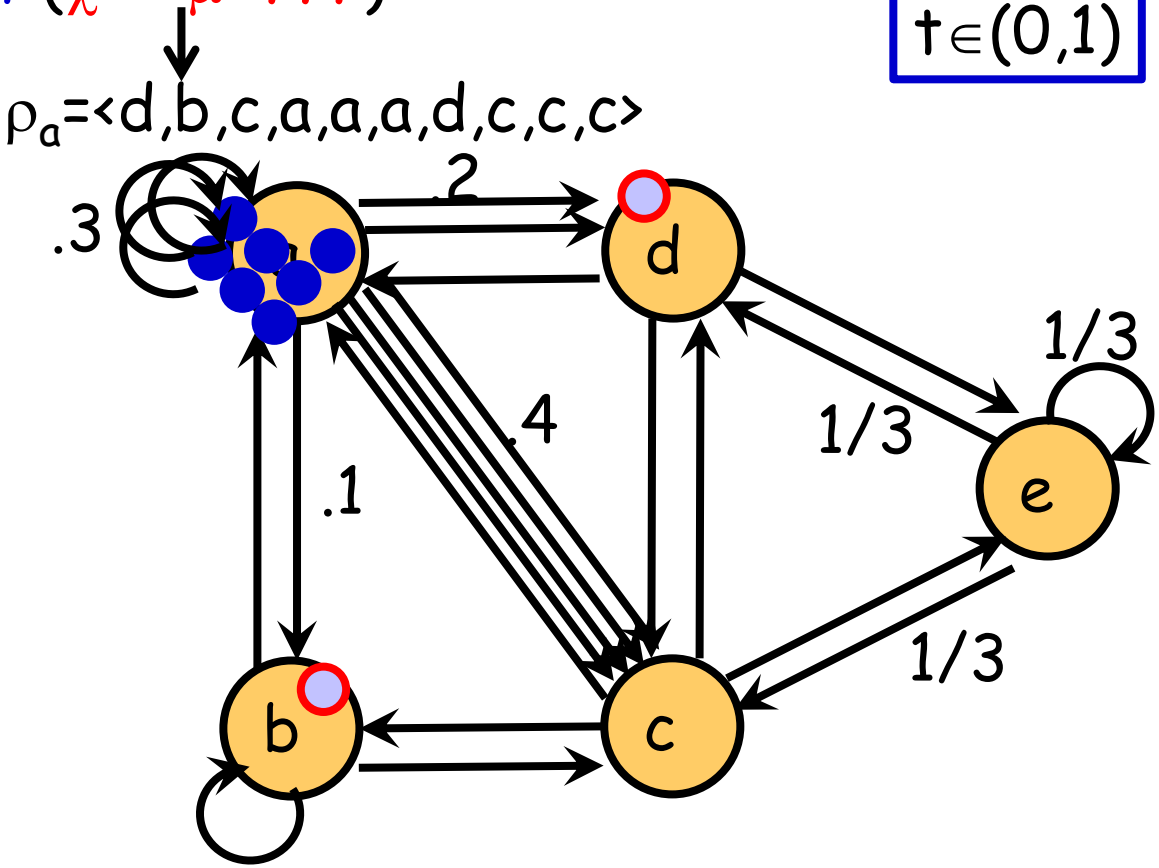


Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

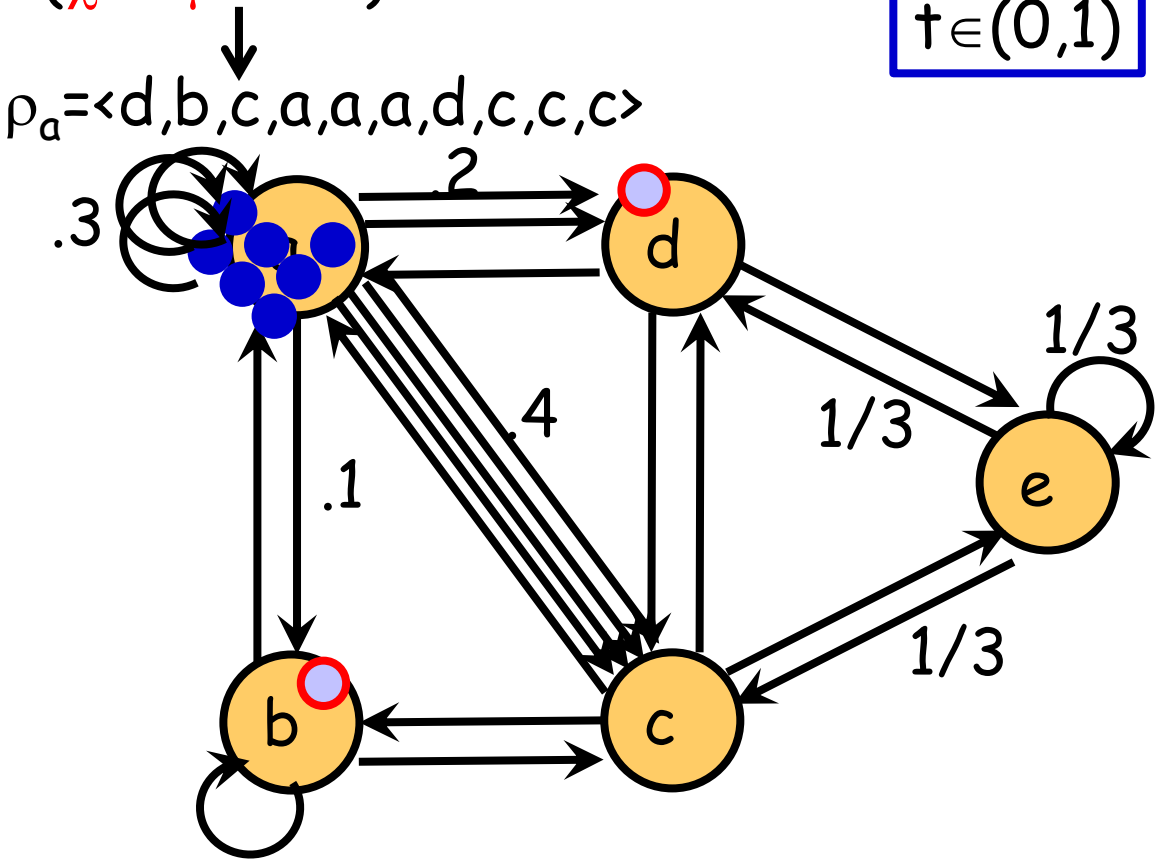


Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

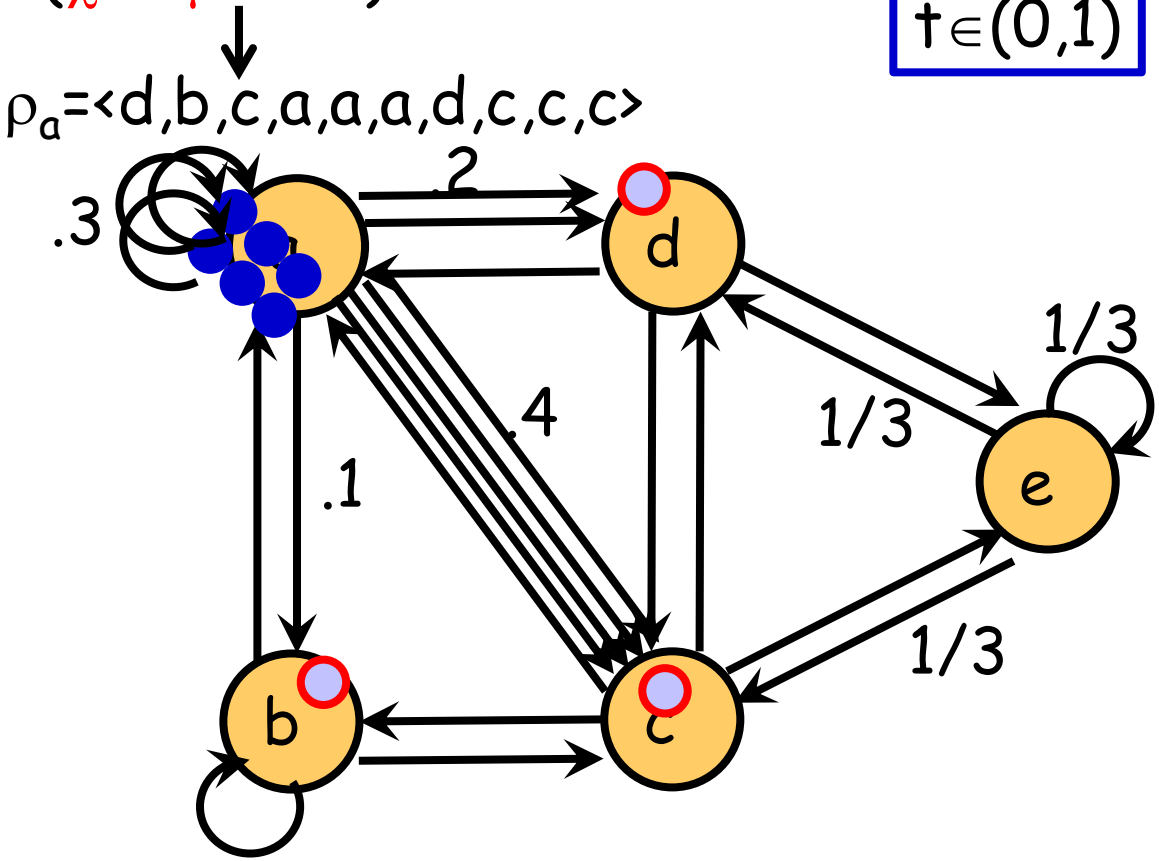


Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

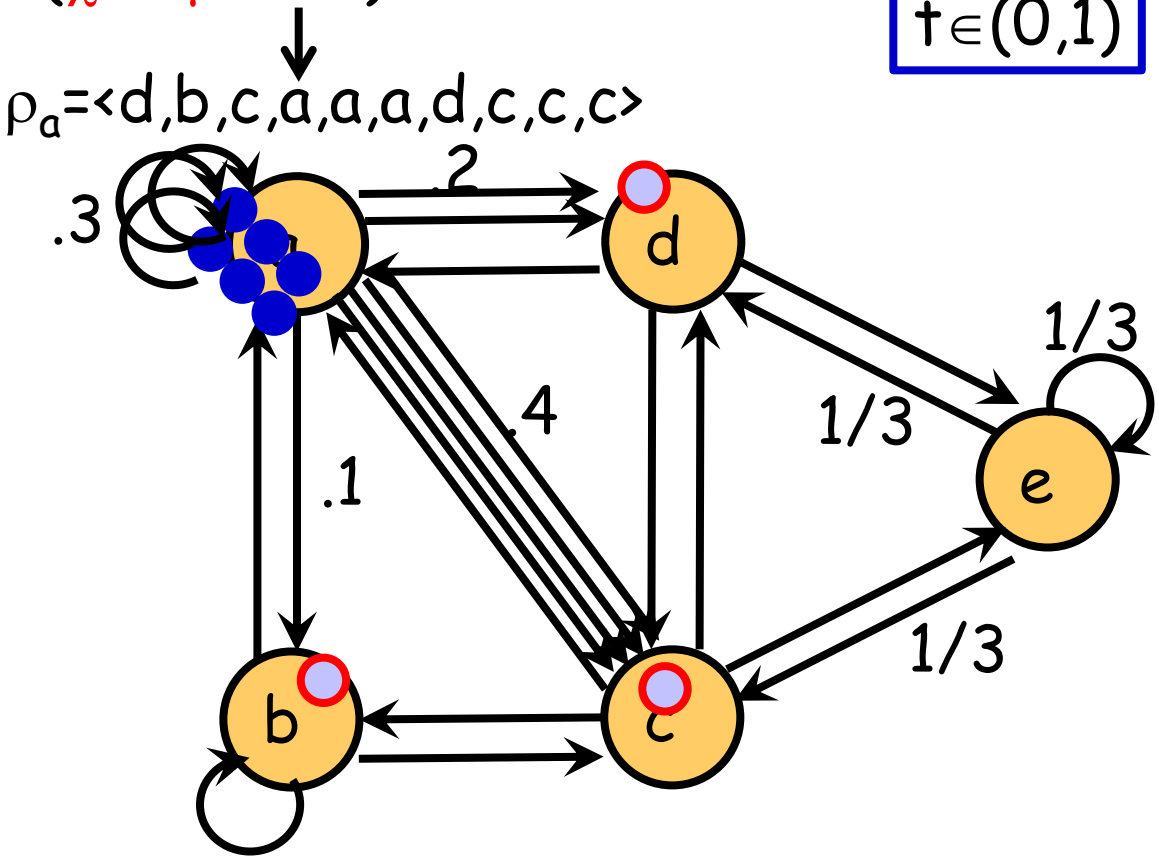


Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

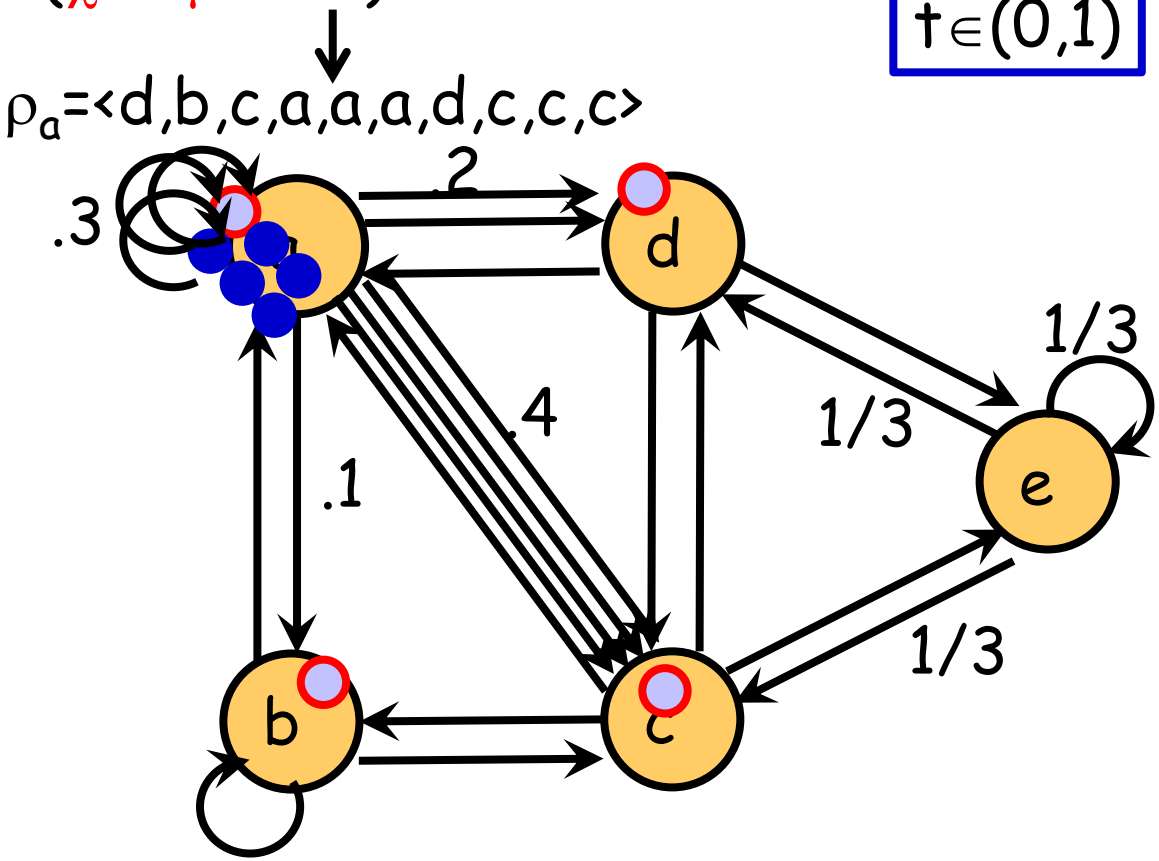


Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

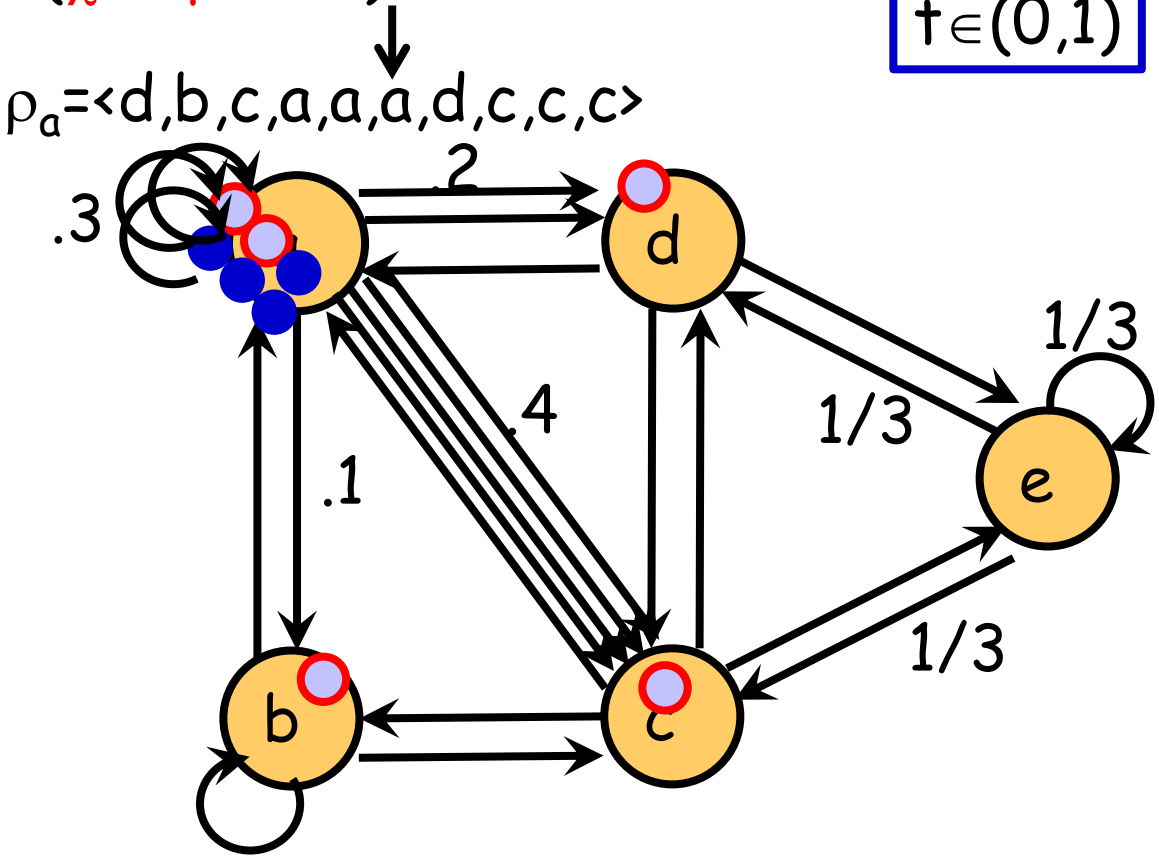


Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

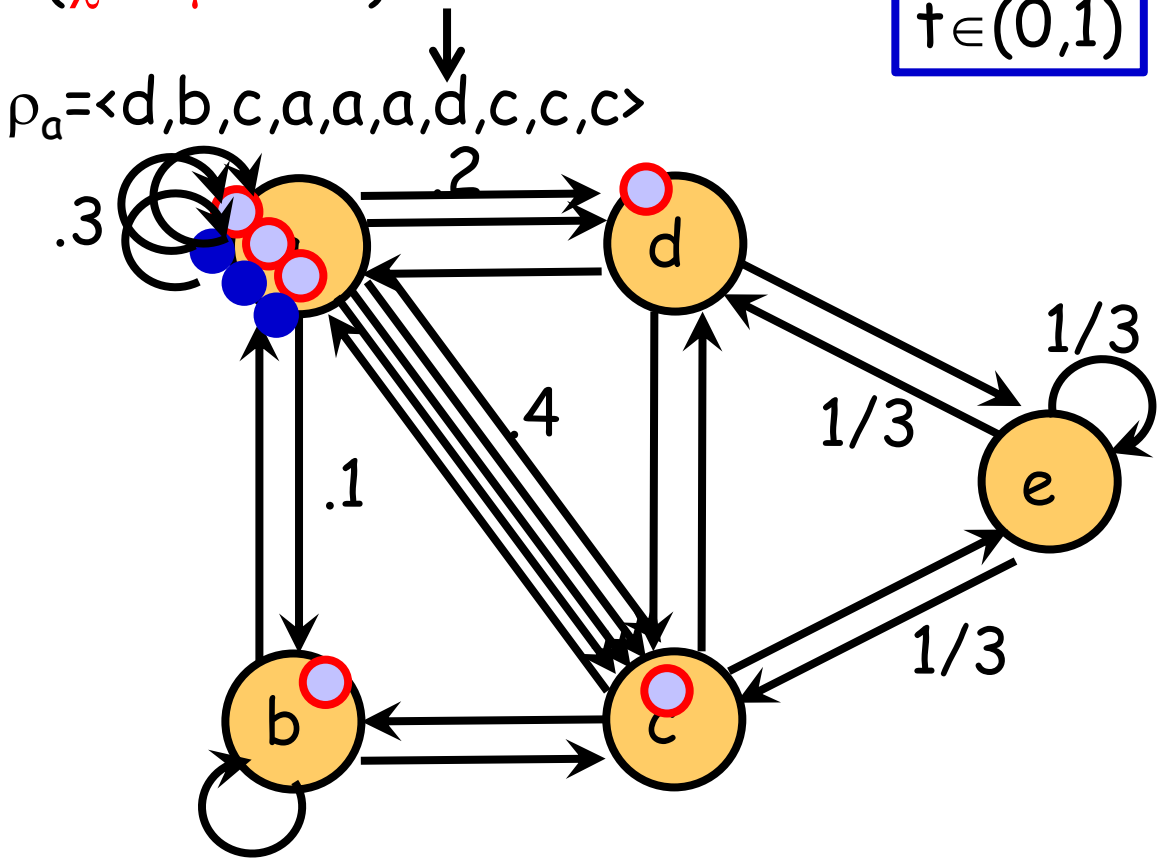


Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

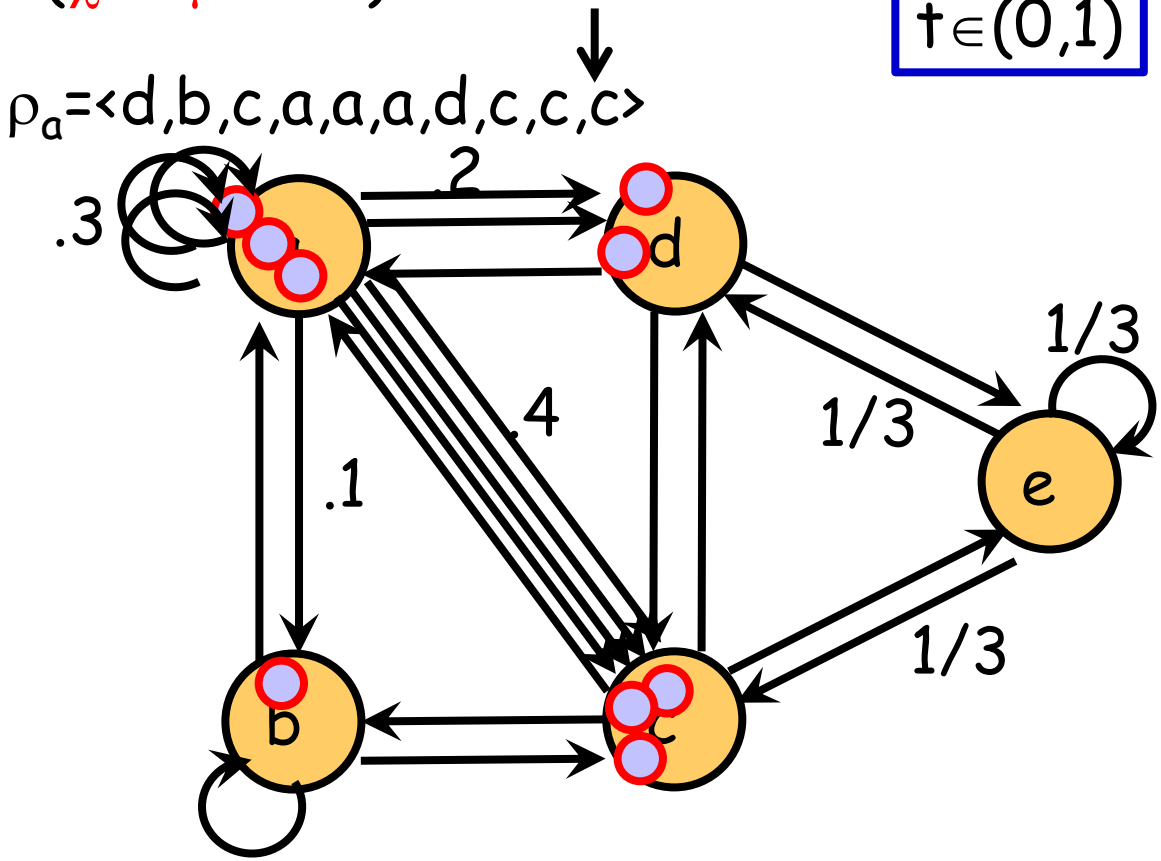


Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

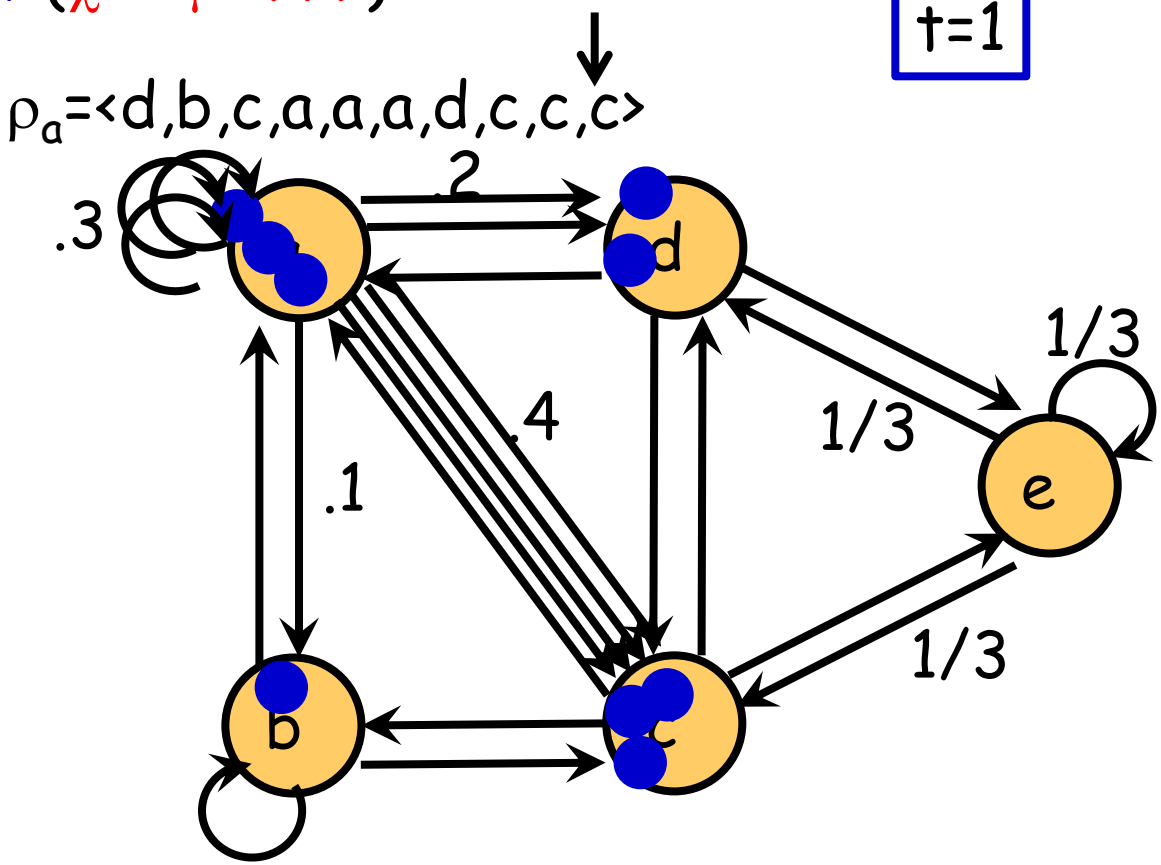
$t \in (0,1)$



Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

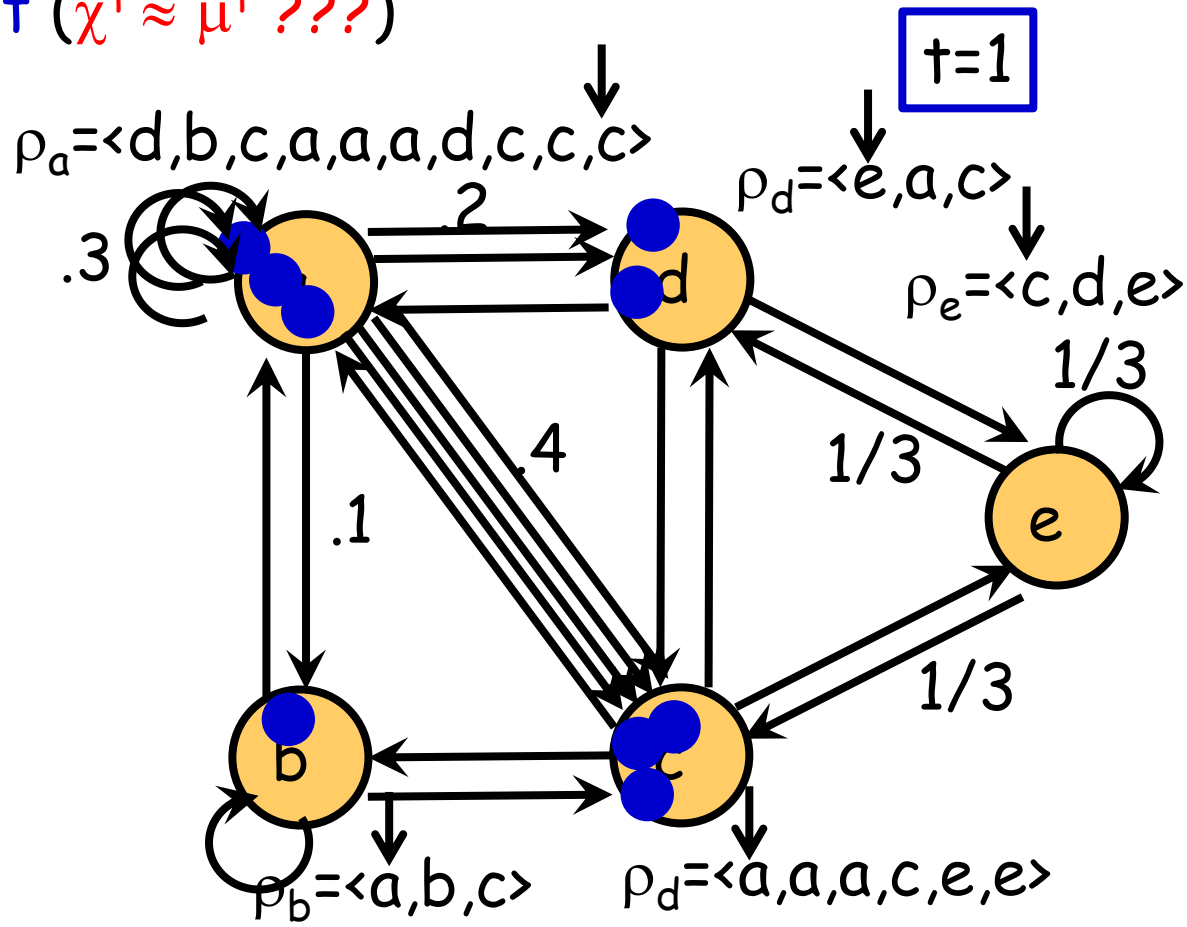
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

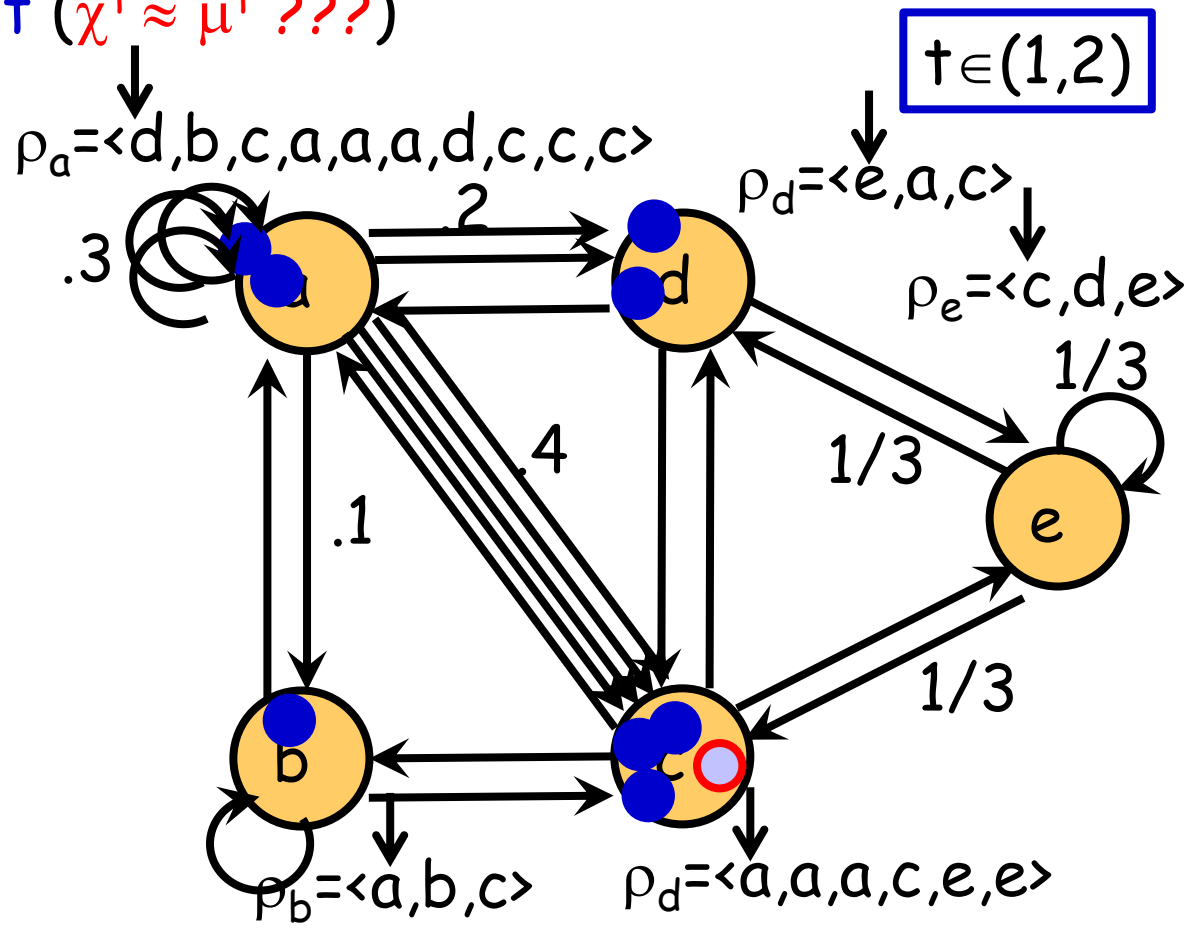
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

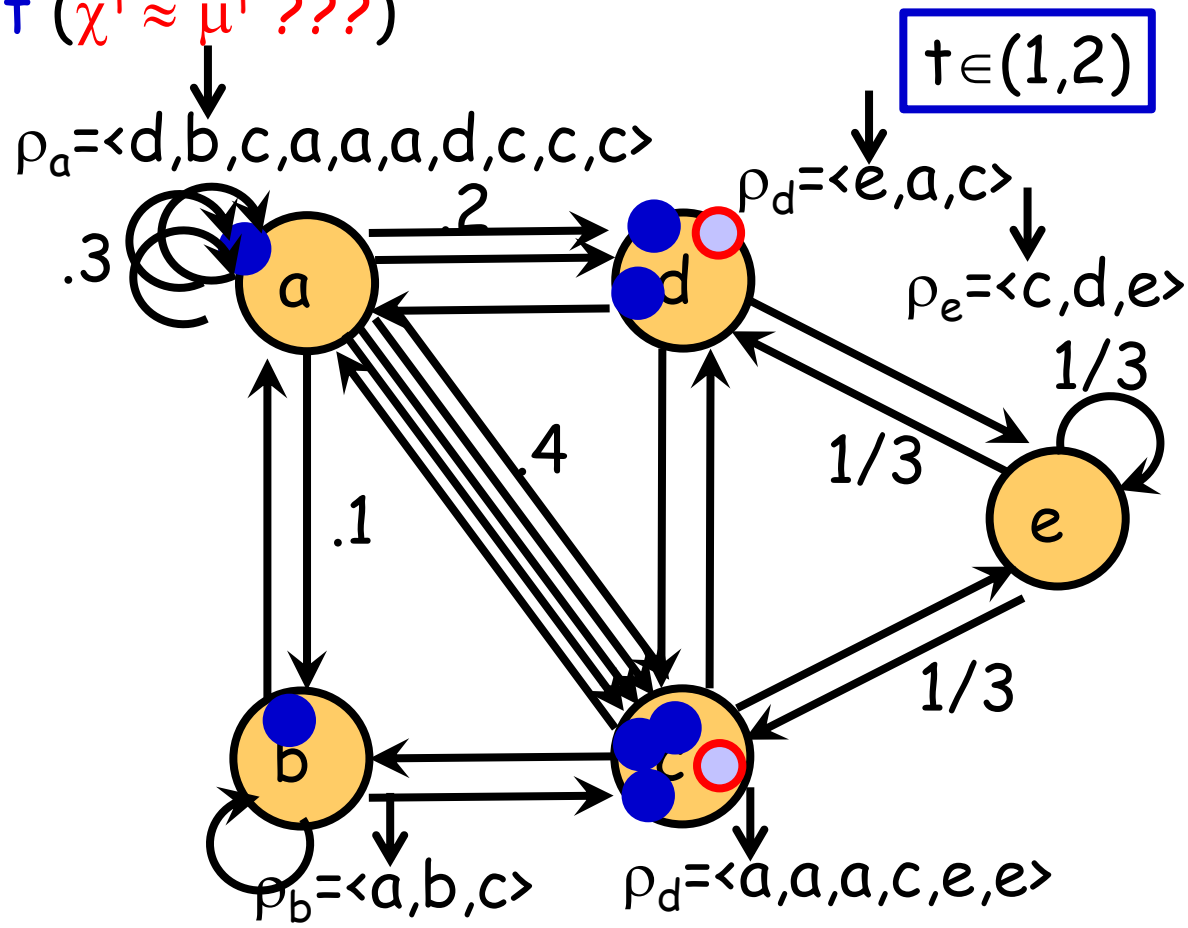
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

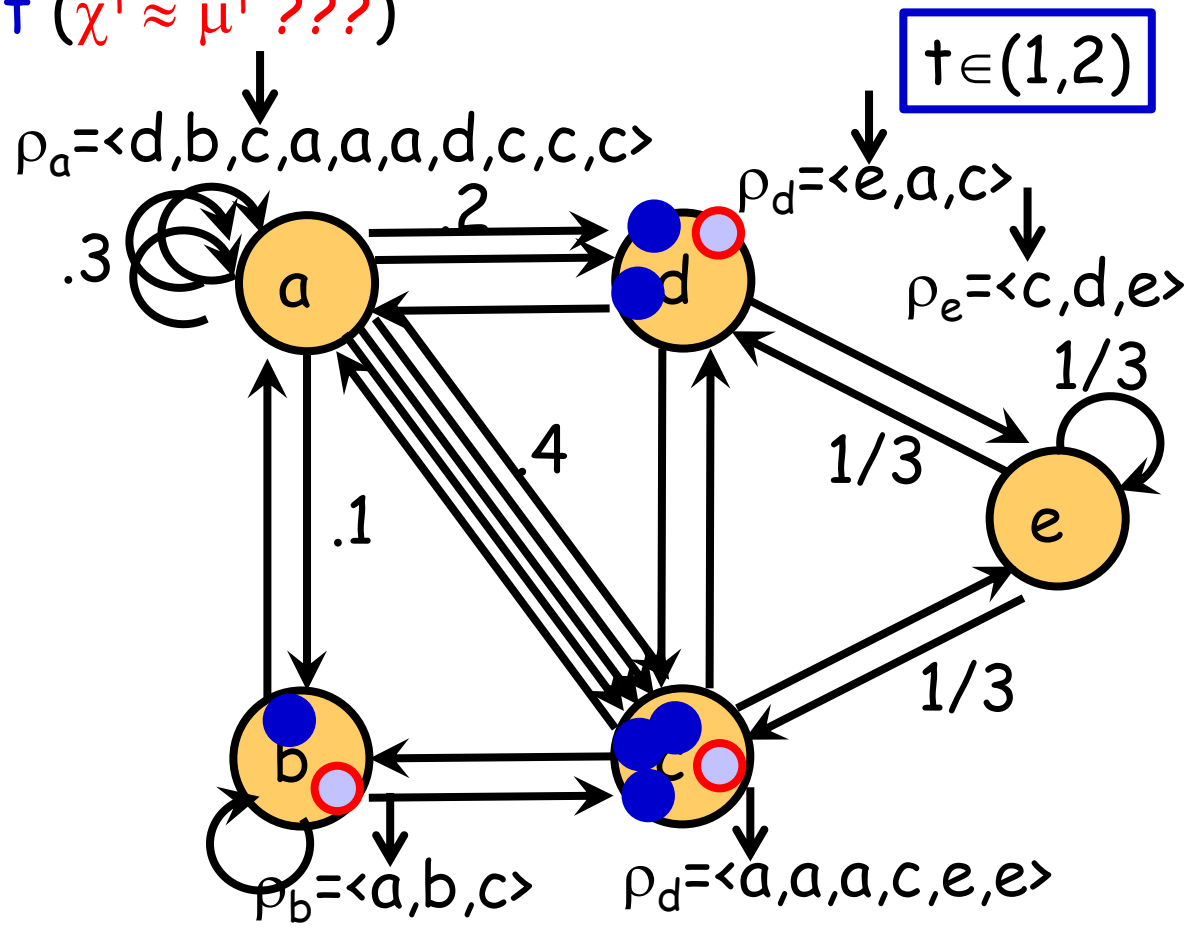
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

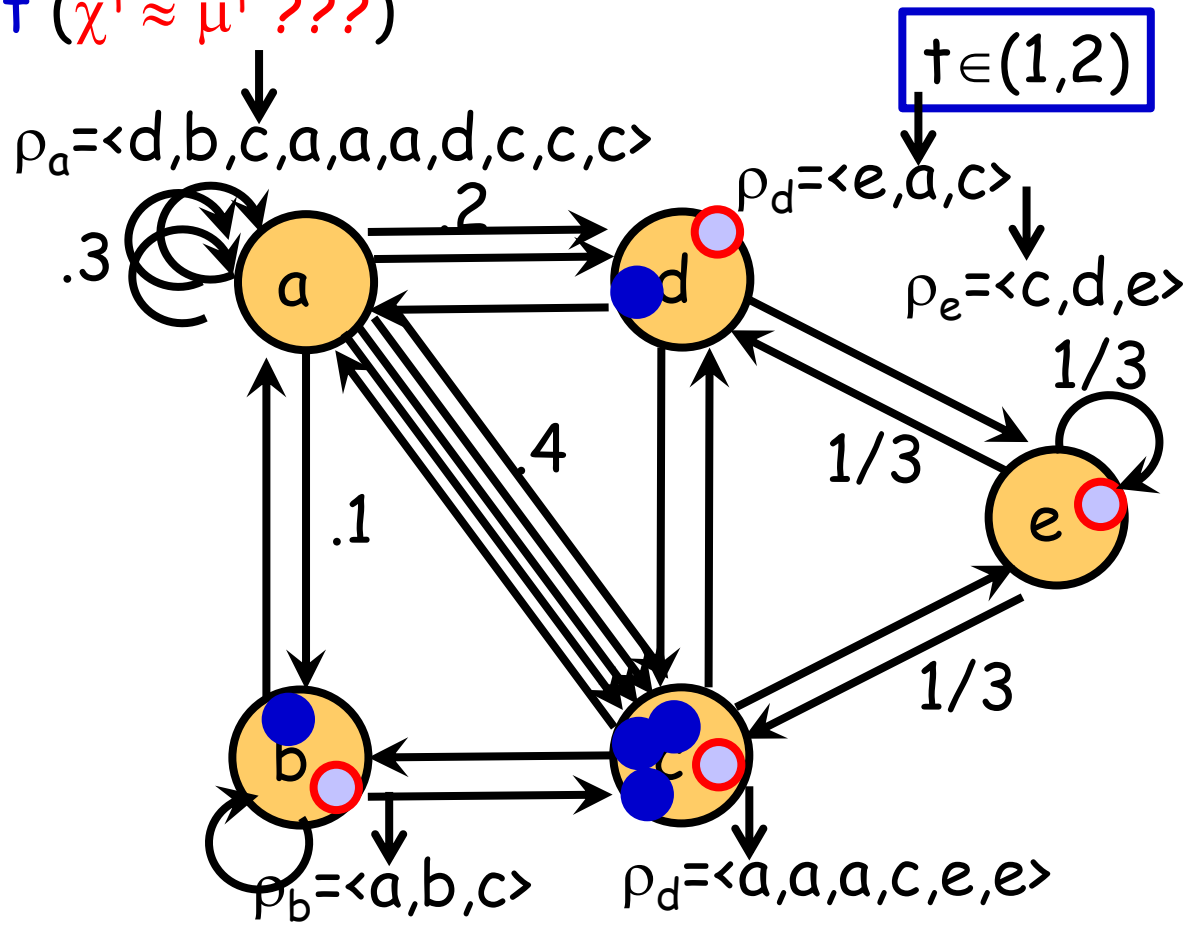
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

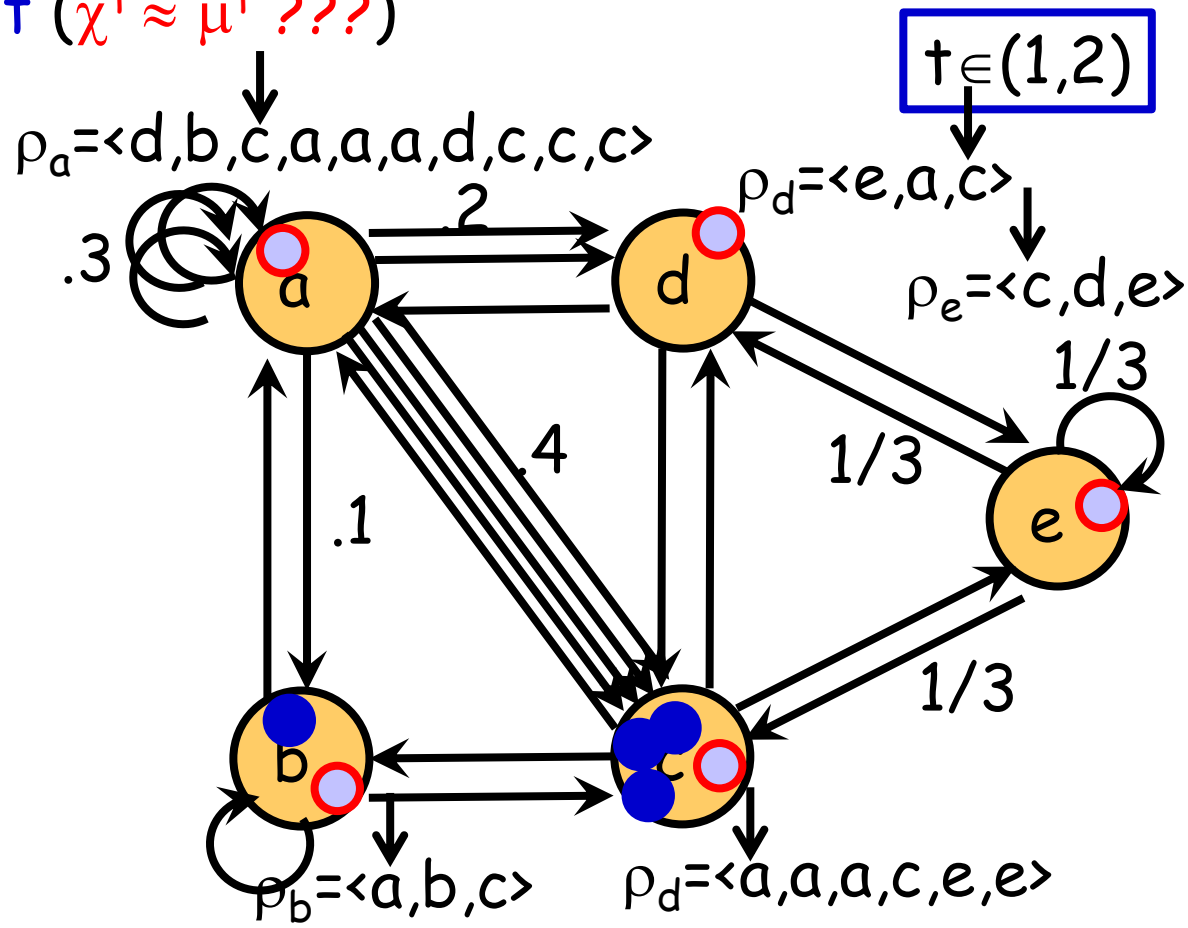
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

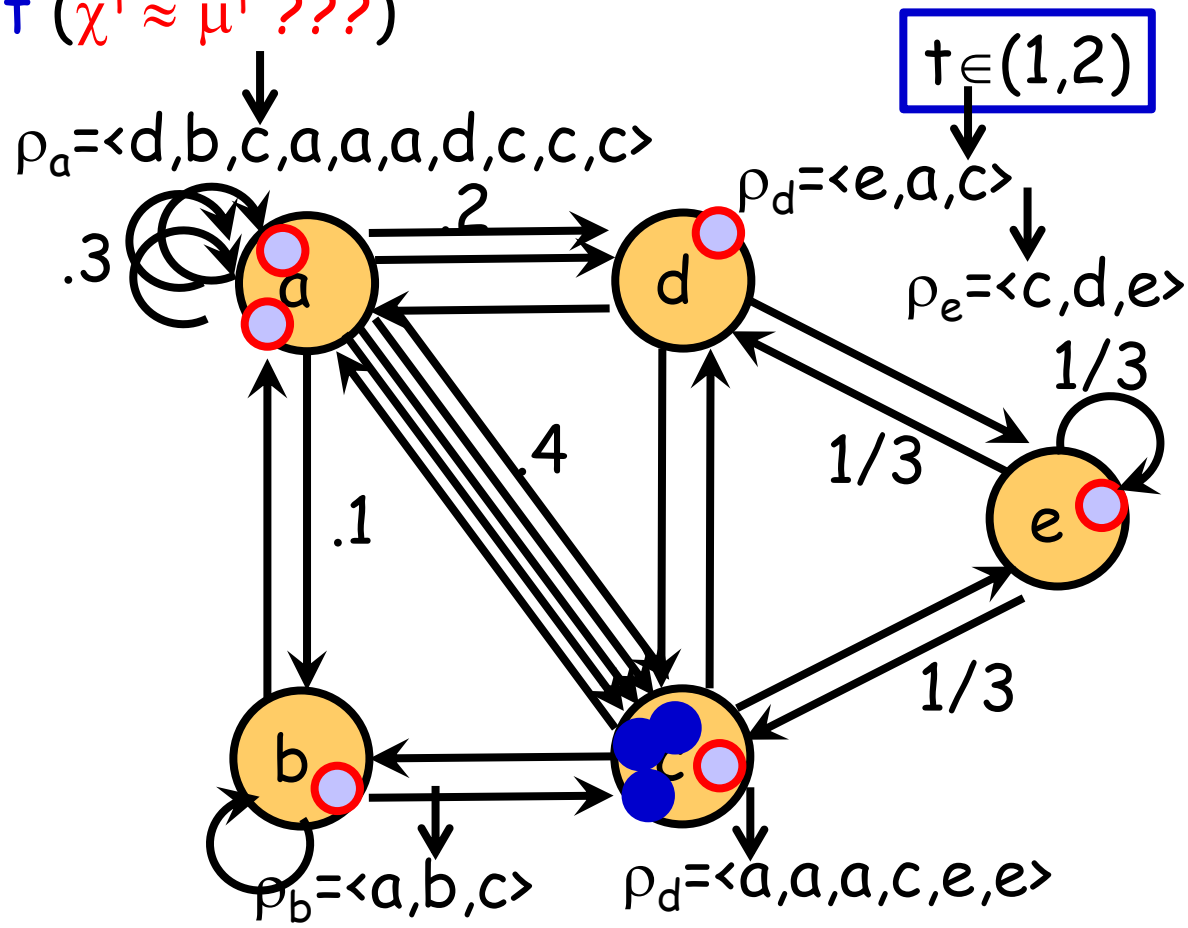
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

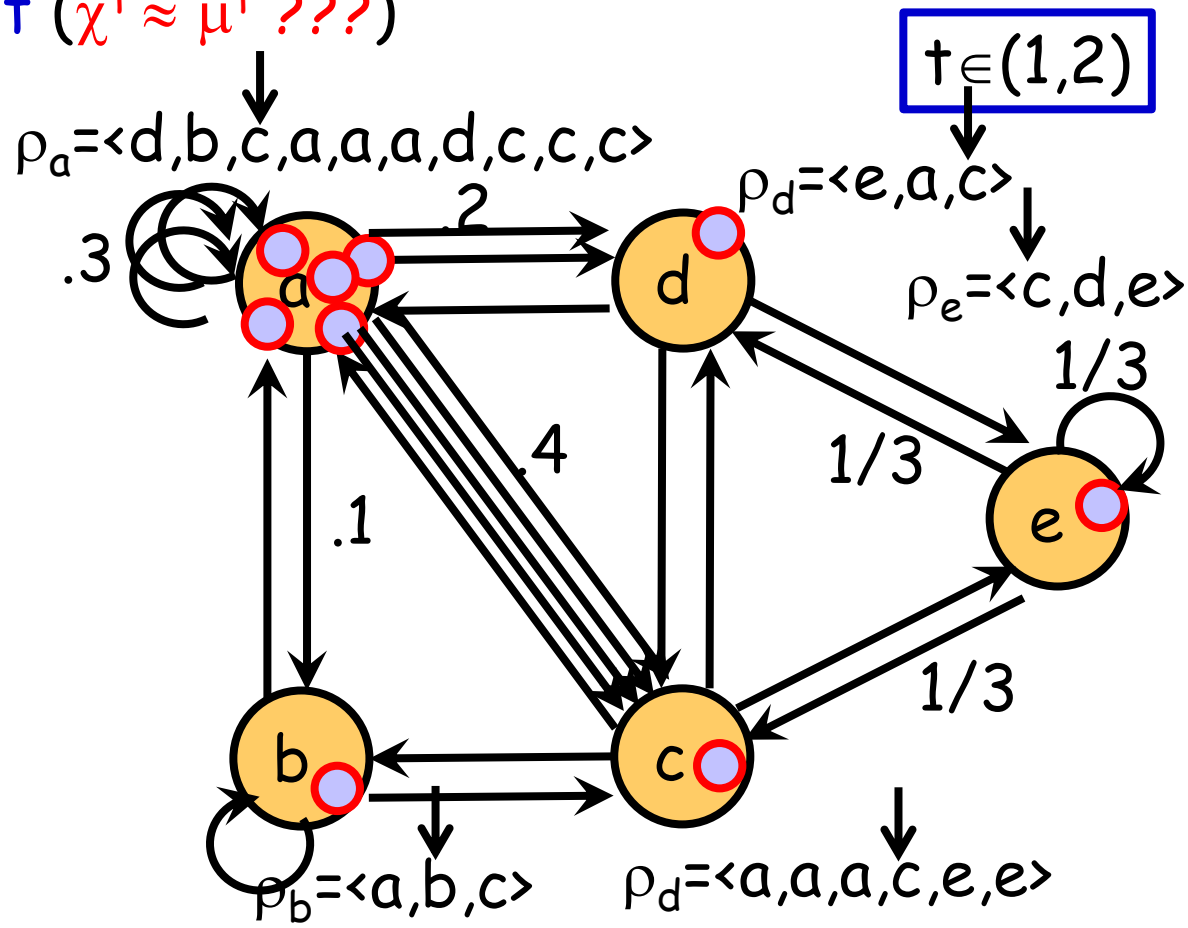
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

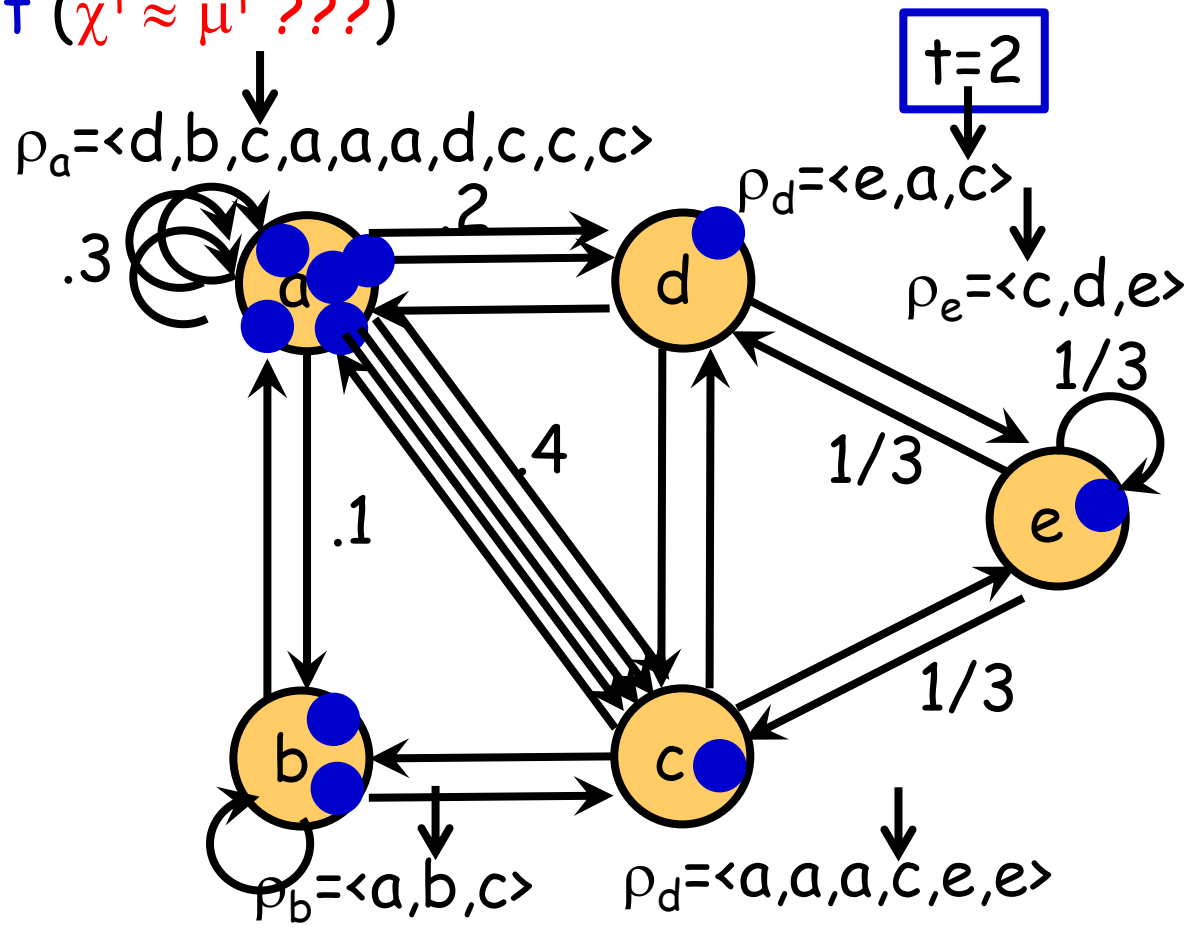
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

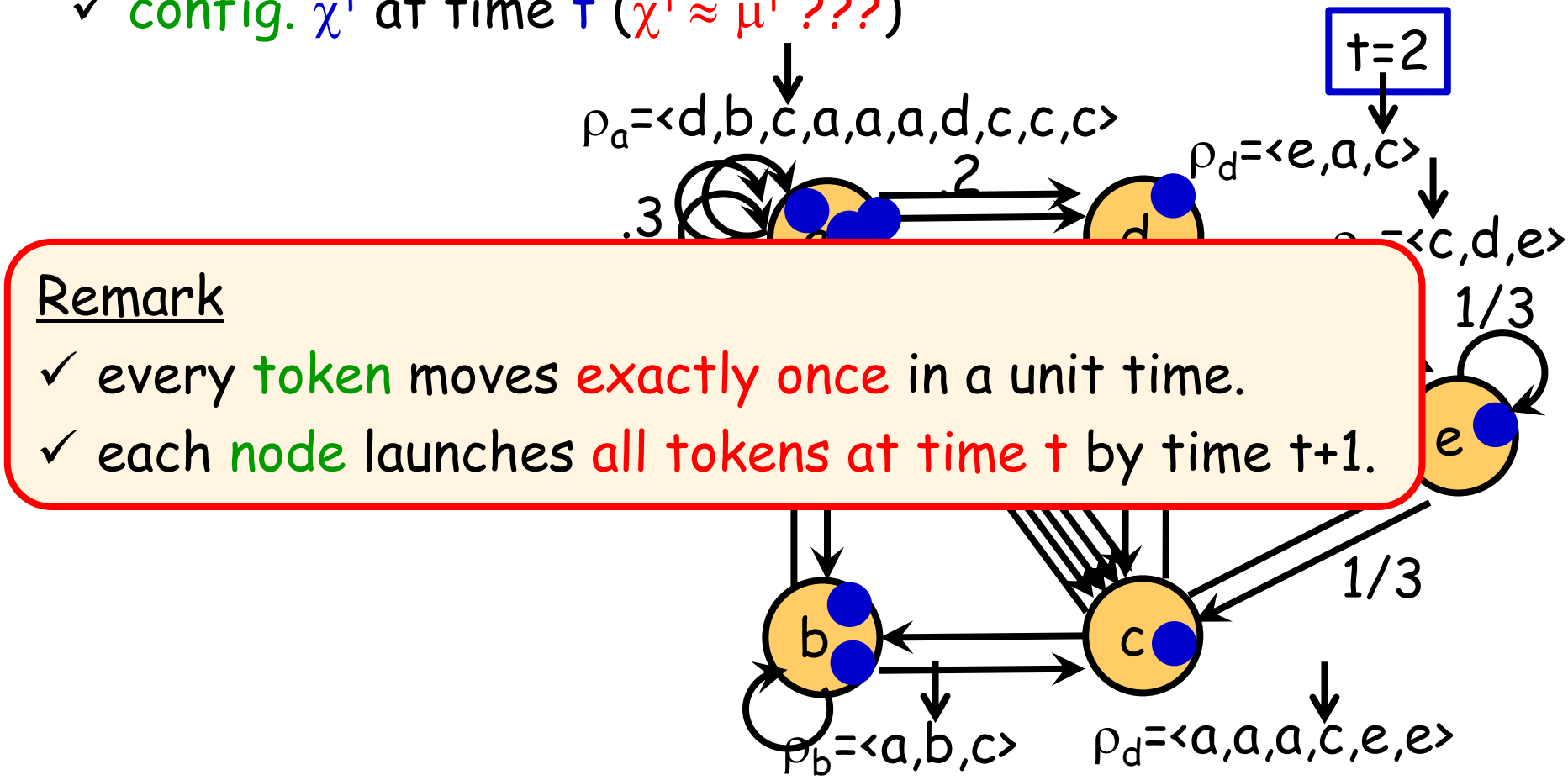
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



Deterministic RW (Propp machine; rotor-router)

M tokens deterministically walks on a graph, by rotor-routers.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ ρ_u: "**rotor router**" on u (launches tokens to v w/ ratio prop. to P_{uv})
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



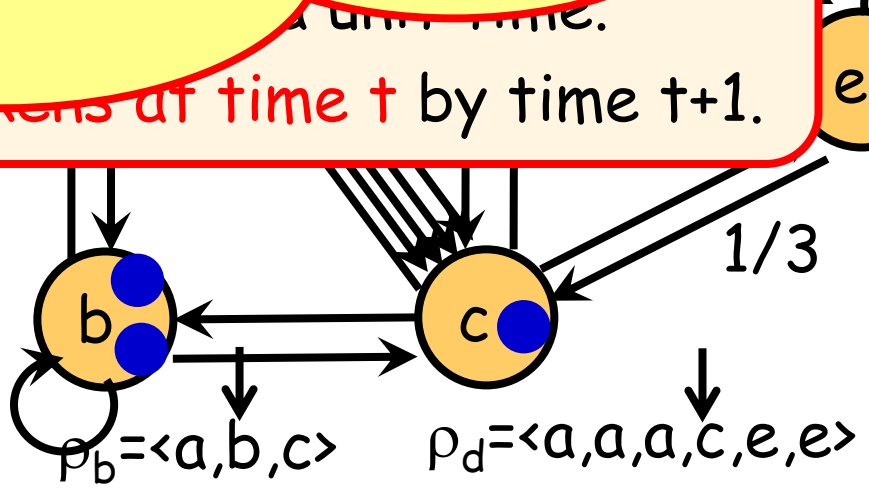
Remark

- ✓ every token moves exactly once in a unit time.
- ✓ each node launches all tokens at time t by time t+1.

Can a det. RW "simulate" a RW?

M tokens deterministically walks on a graph, by rotor-routers.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ to P_{uv})
- ✓ (ergodic) Markov chain converges to the stat. distr.
  - ➢ tokens "mix" by RW.
- ✓ Can the Propp machine **simulate random walk**?
  - ➢ Do tokens "mix" by det RW?
    i.e., No "lumps"?
- ✓ ev… a uni… time.
- ✓ each node launches all tokens at time t by time t+1.



Main Results (1/3)

about lower bound of single vertex discrepancy

Thm. [K, Koga, Makino 10+]

There exists a multidigraph $\mathcal{G}$=(V,$\mathcal{E}$),

exists ini. config, and exists rotor-router, such that

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right| \geq \Omega(m)$$

i.e., "lump" w/ a size Ω(m)

holds, where m = $|\mathcal{E}|$,

$\chi_w^{(T)}$: #tokens @ w∈V, @ time T > 0 in det. RW

$\mu_w^{(T)}$: E[#tokens] @ w∈V , @ time T > 0 in RW

An example of "lumps"

Thm. [K, Koga, Makino 10+]

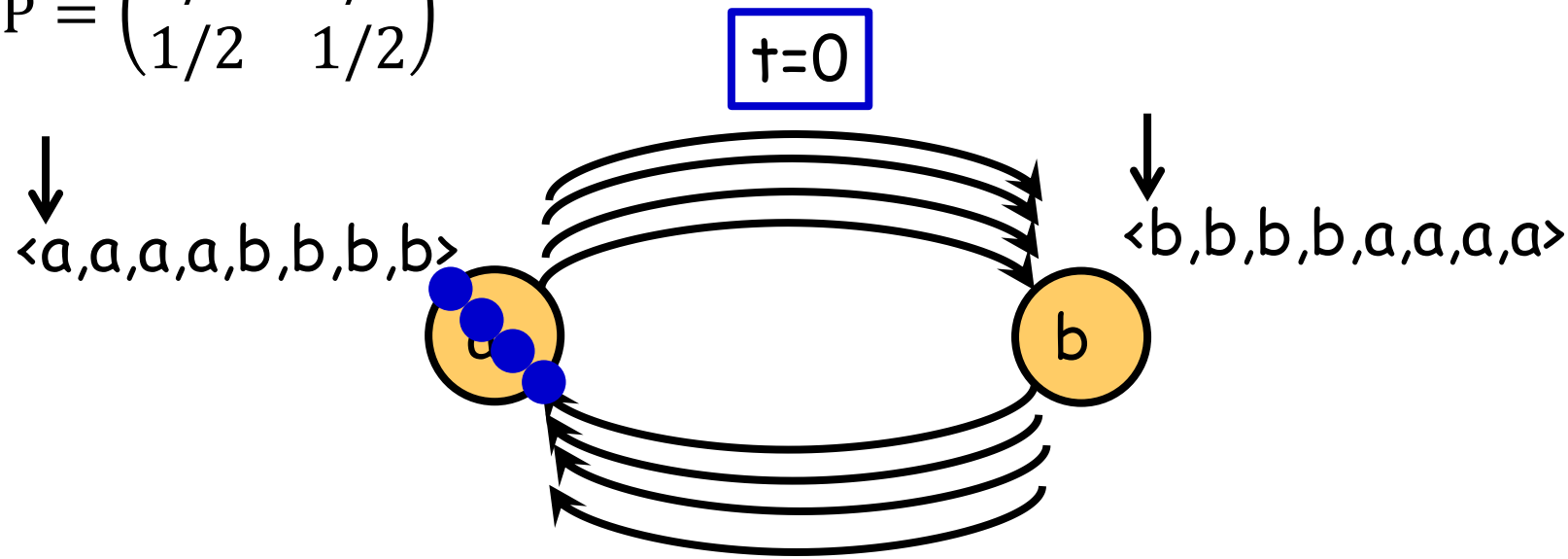
There exists a multidigraph $\mathcal{G}=(V,\mathcal{E})$,

exists ini. config, and exists rotor-router, such that

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right| \geq \Omega(m)$$

Rem: corresp. trans. matrix

$$\mathrm{P} = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$



An example of "lumps"

Thm. [K, Koga, Makino 10+]

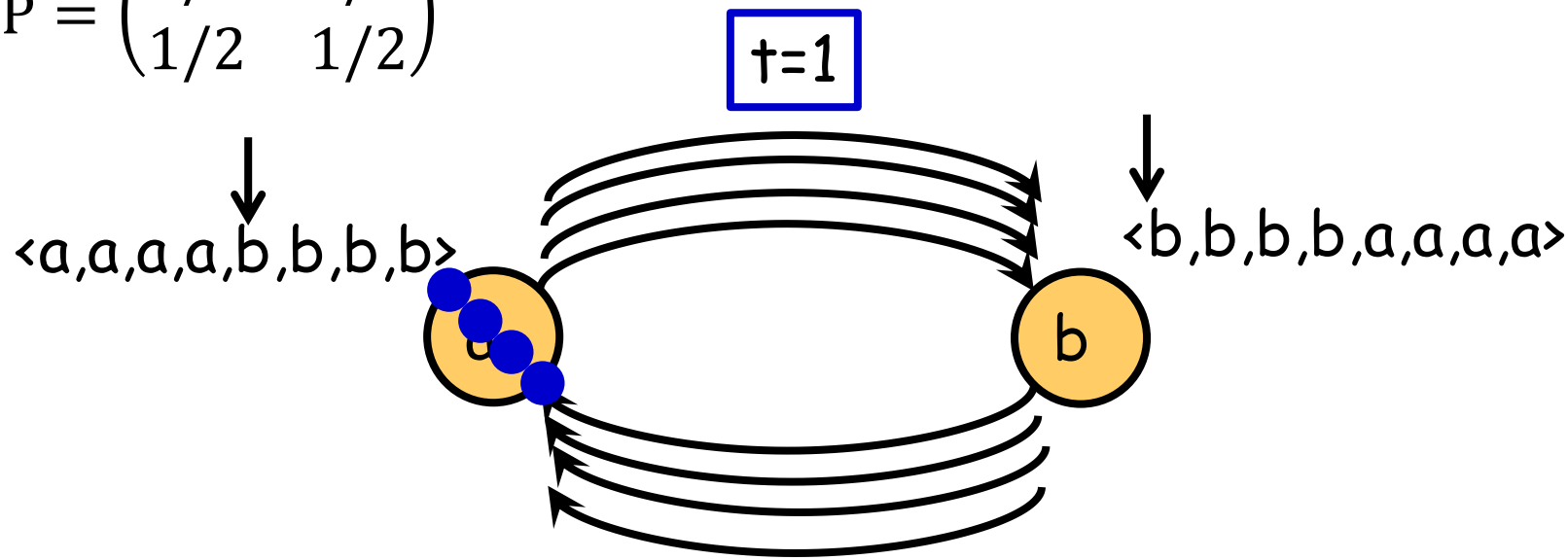
There exists a multidigraph $\mathcal{G}=(V,\mathcal{E})$,

exists ini. config, and exists rotor-router, such that

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right| \geq \Omega(m)$$

Rem: corresp. trans. matrix

$$\mathrm{P} = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$



An example of "lumps"

Thm. [K, Koga, Makino 10+]

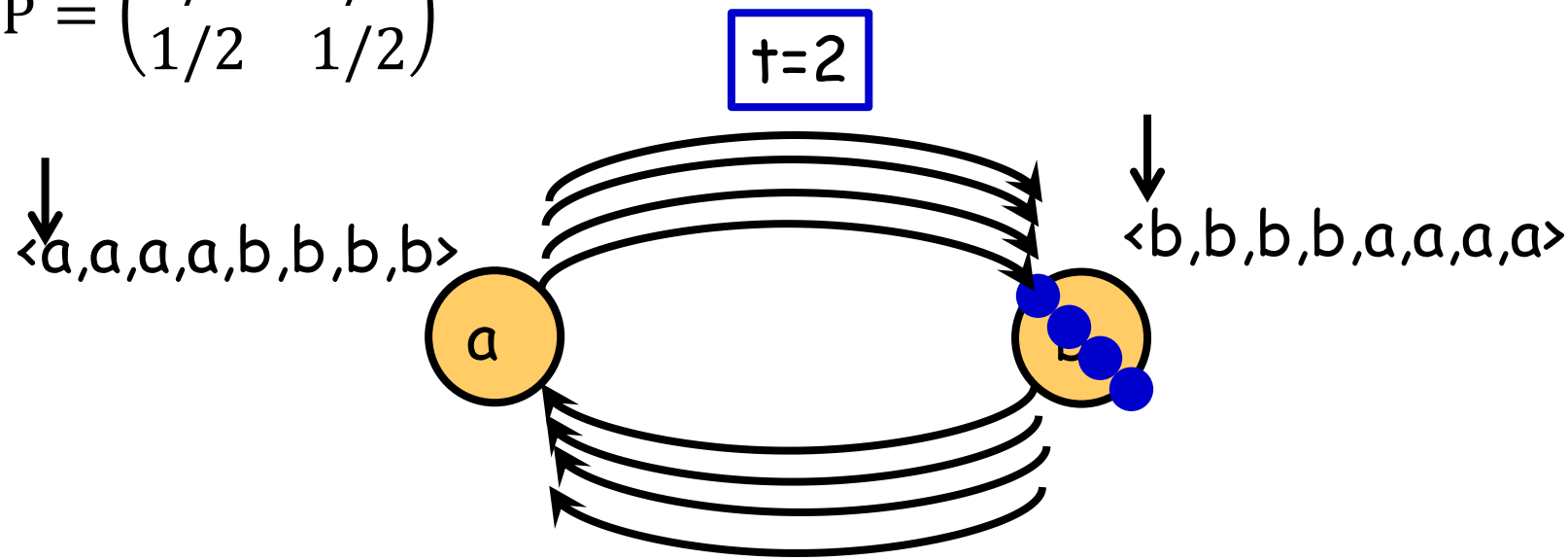
There exists a multidigraph $\mathcal{G}=(V,\mathcal{E})$,

exists ini. config, and exists rotor-router, such that

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right| \geq \Omega(m)$$

Rem: corresp. trans. matrix

$$\mathrm{P} = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$



An example of "lumps"

Thm. [K, Koga, Makino 10+]

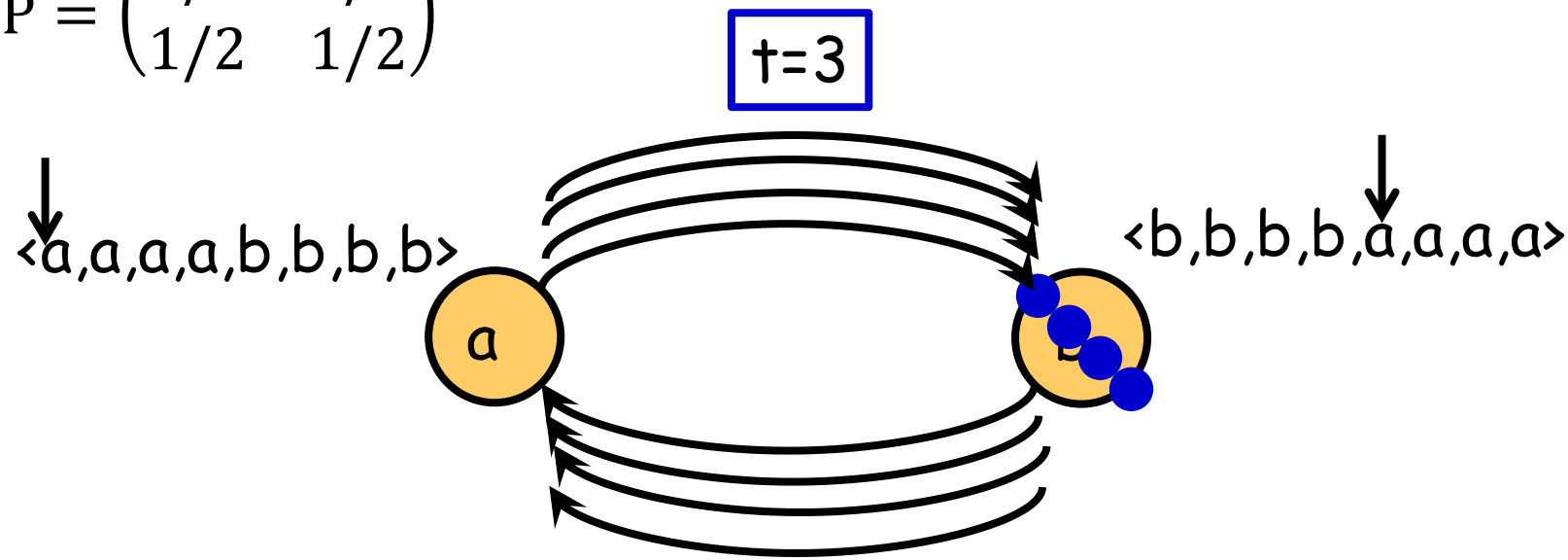
There exists a multidigraph $\mathcal{G}=(V,\mathcal{E})$,

exists ini. config, and exists rotor-router, such that

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right| \geq \Omega(m)$$

Rem: corresp. trans. matrix

$$\mathrm{P} = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$



An example of "lumps"

Thm. [K, Koga, Makino 10+]

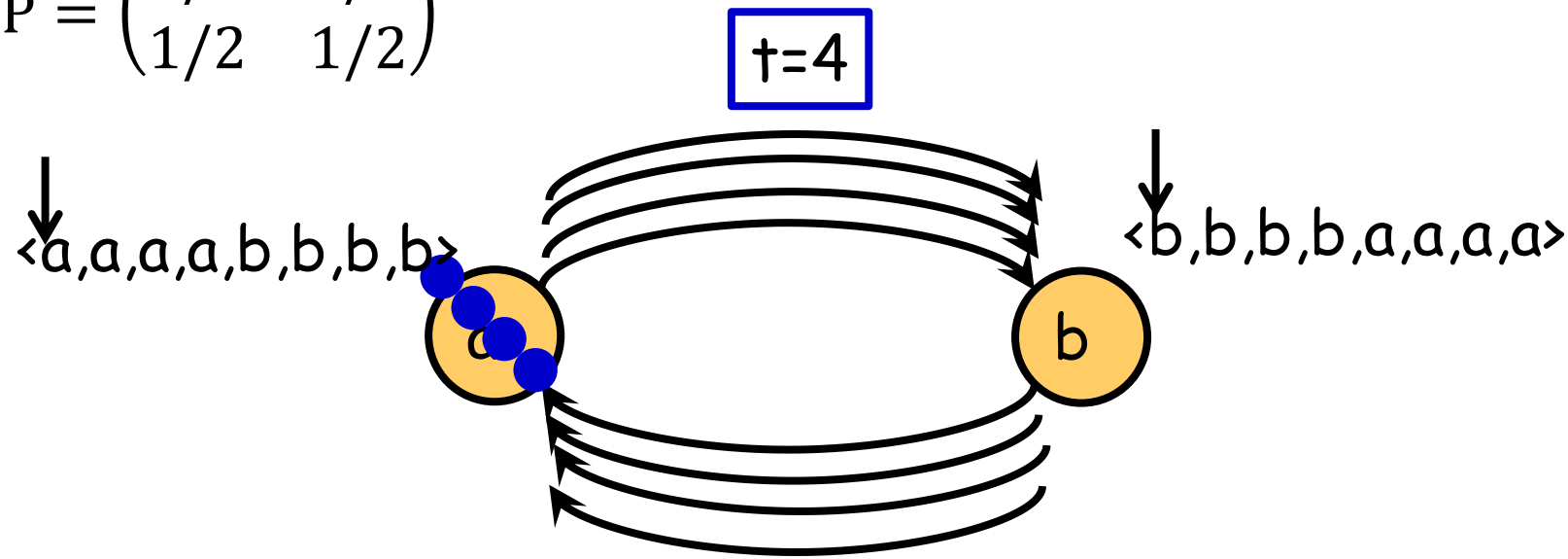
There exists a multidigraph $\mathcal{G}=(V,\mathcal{E})$,

exists ini. config, and exists rotor-router, such that

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right| \geq \Omega(m)$$

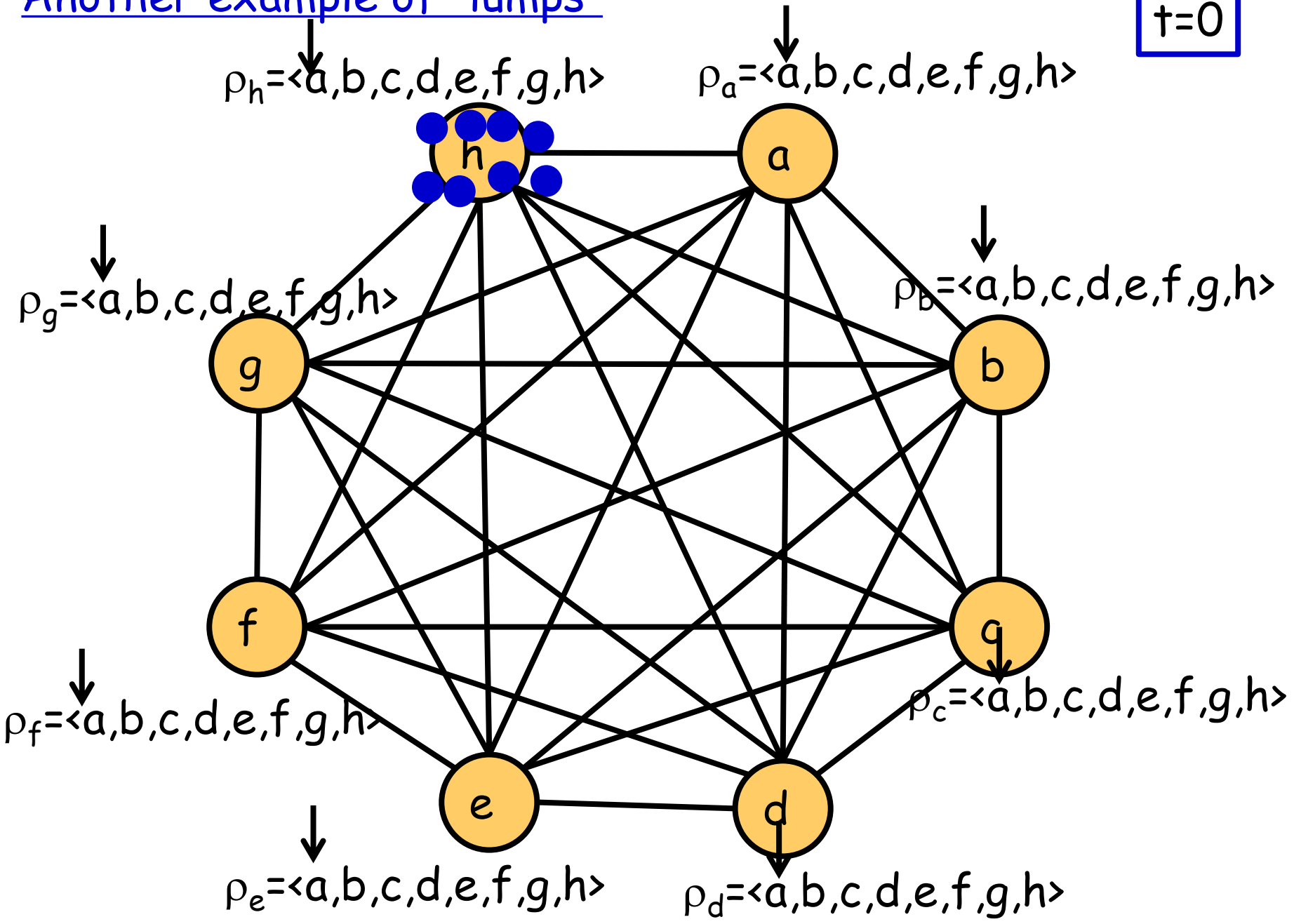
Rem: corresp. trans. matrix

$$\mathrm{P} = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$



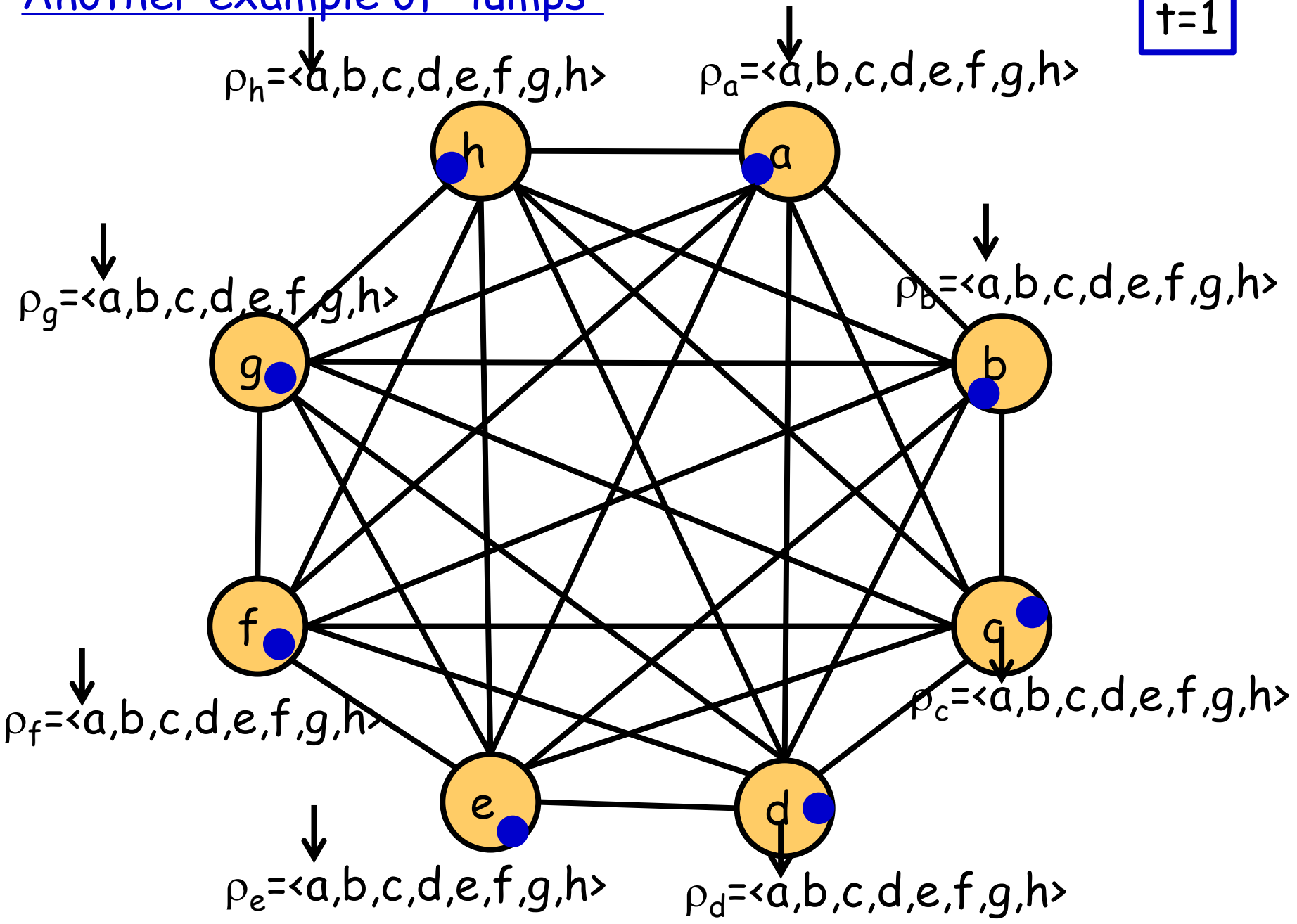
Another example of "lumps"

t=0



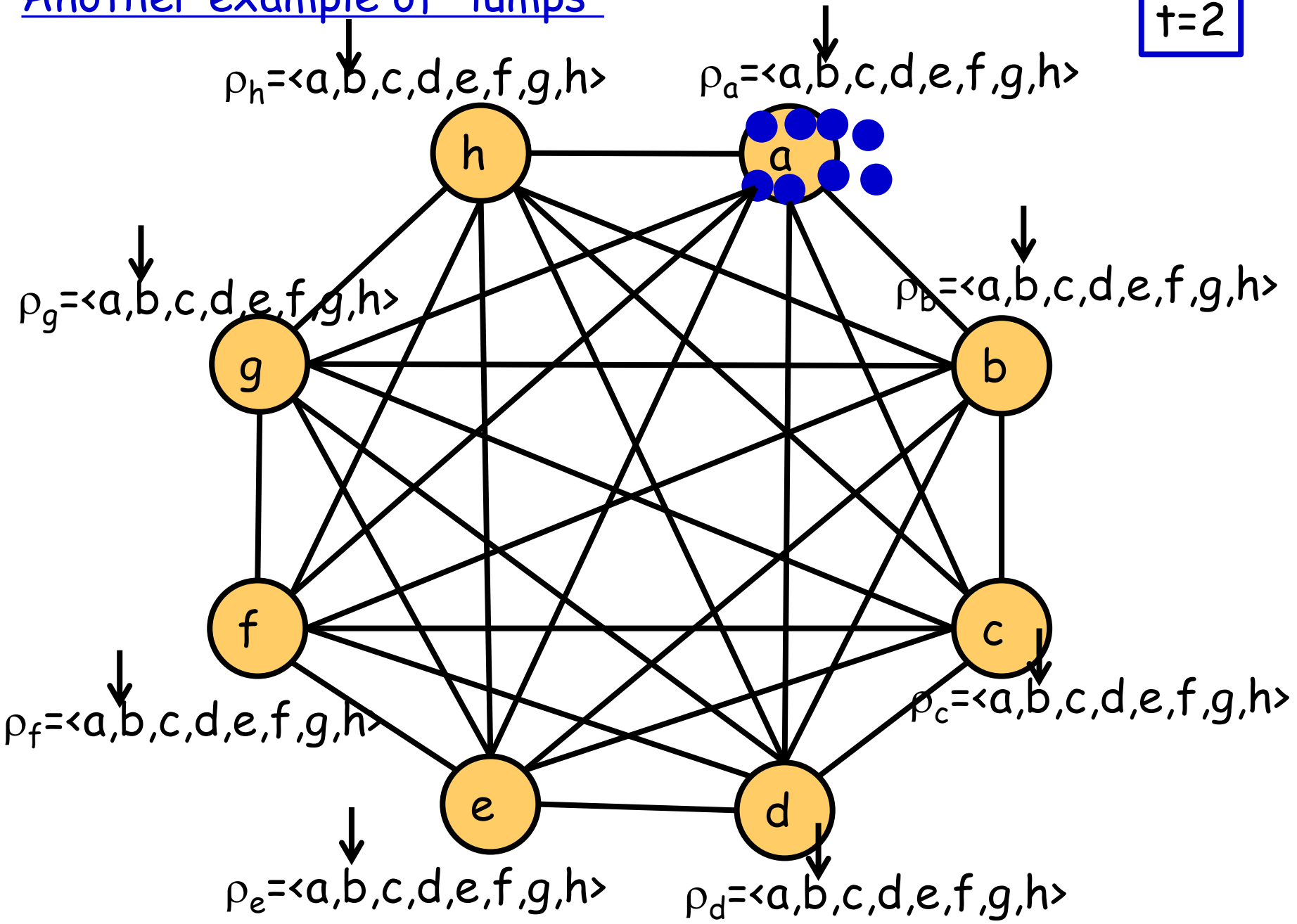
Another example of "lumps"

t=1



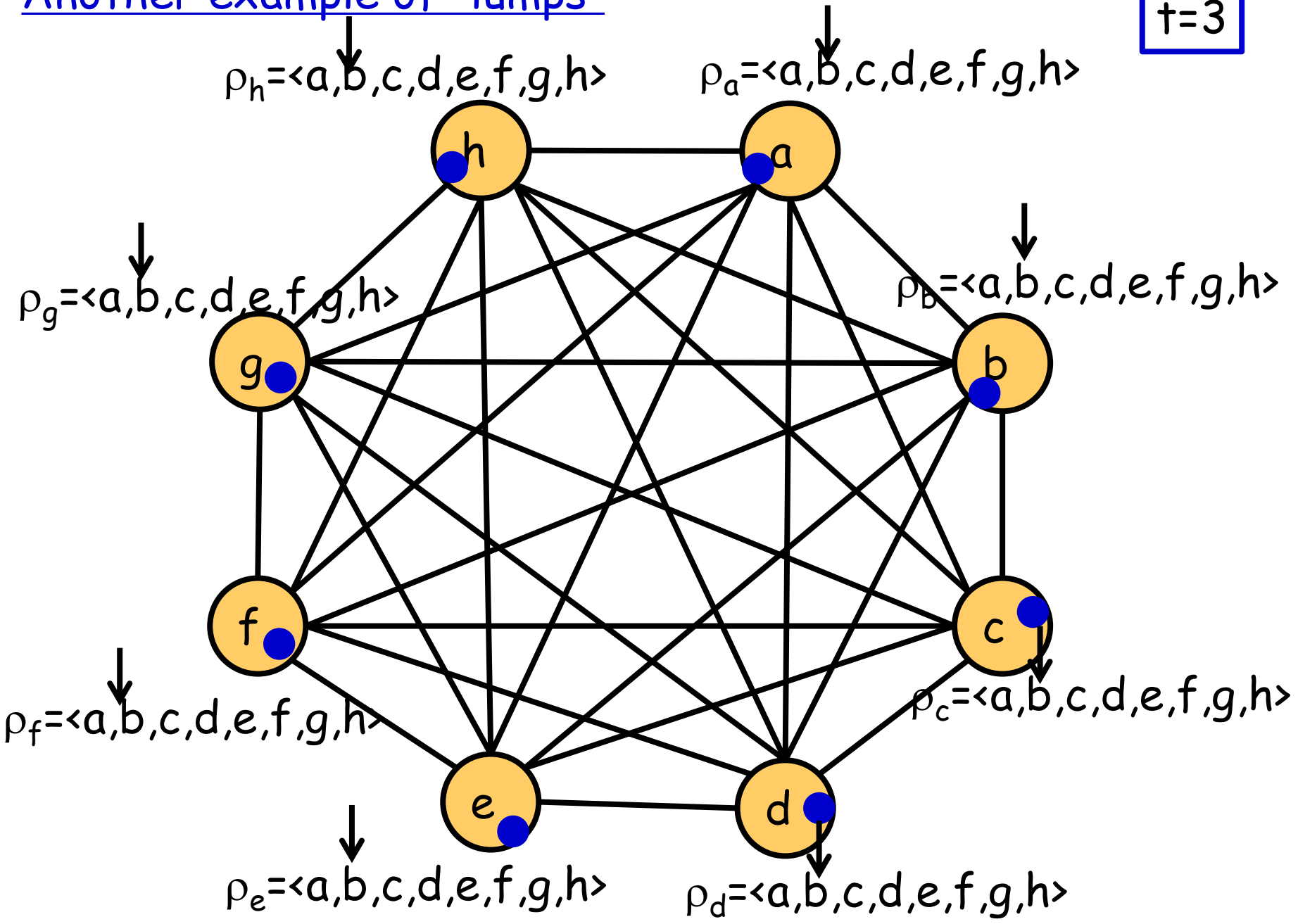
Another example of "lumps"

t=2



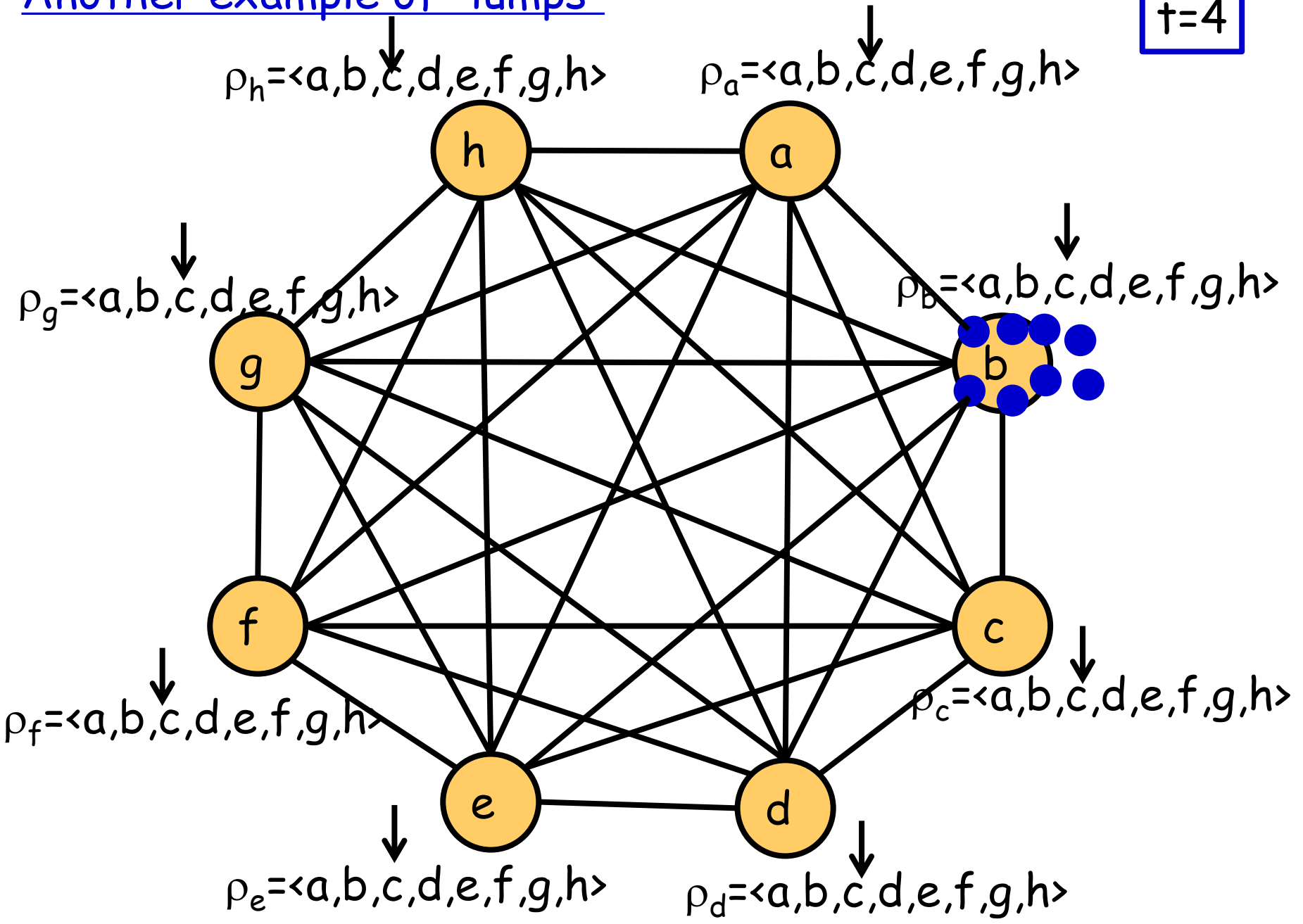
Another example of "lumps"

t=3



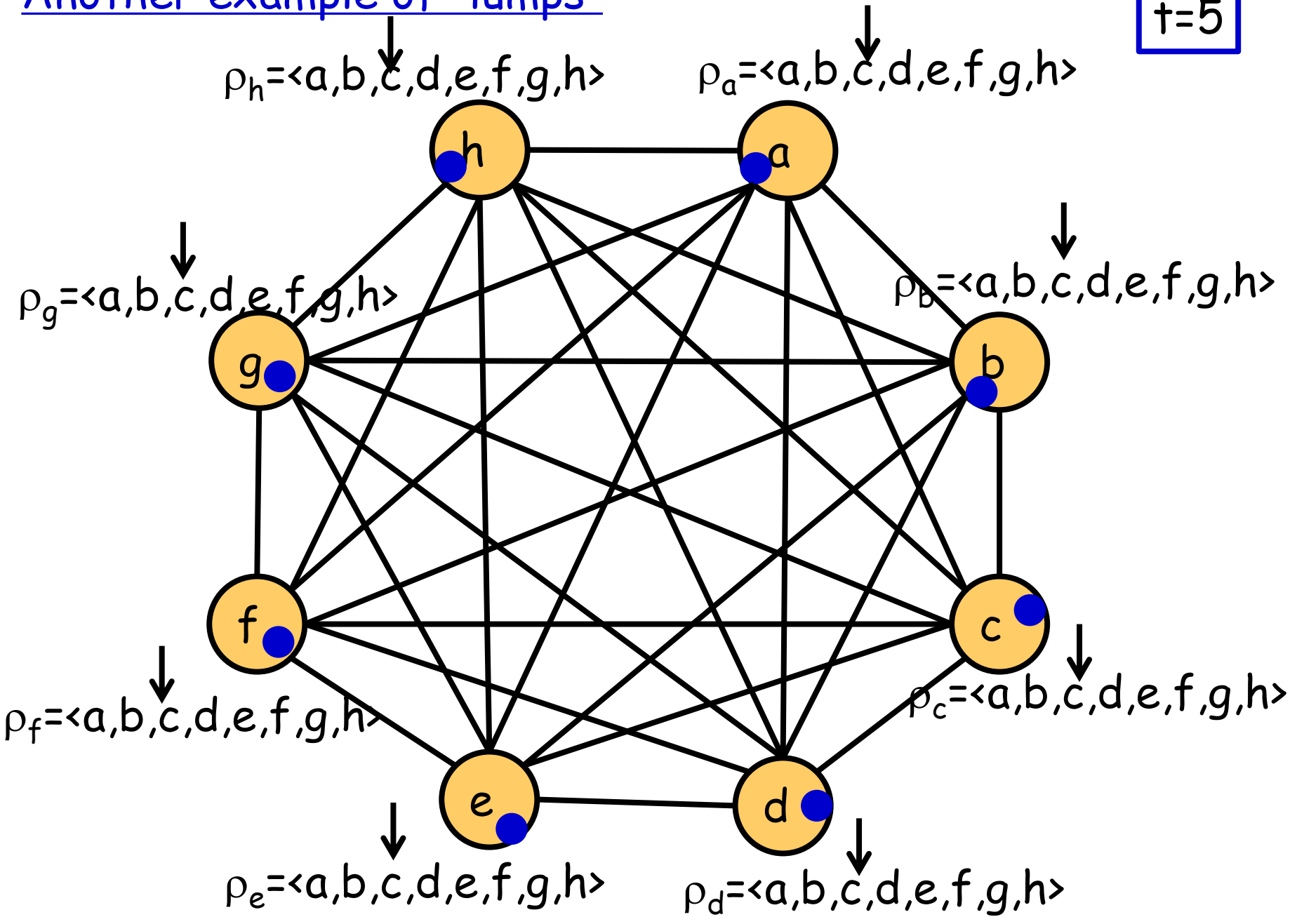
Another example of "lumps"

t=4



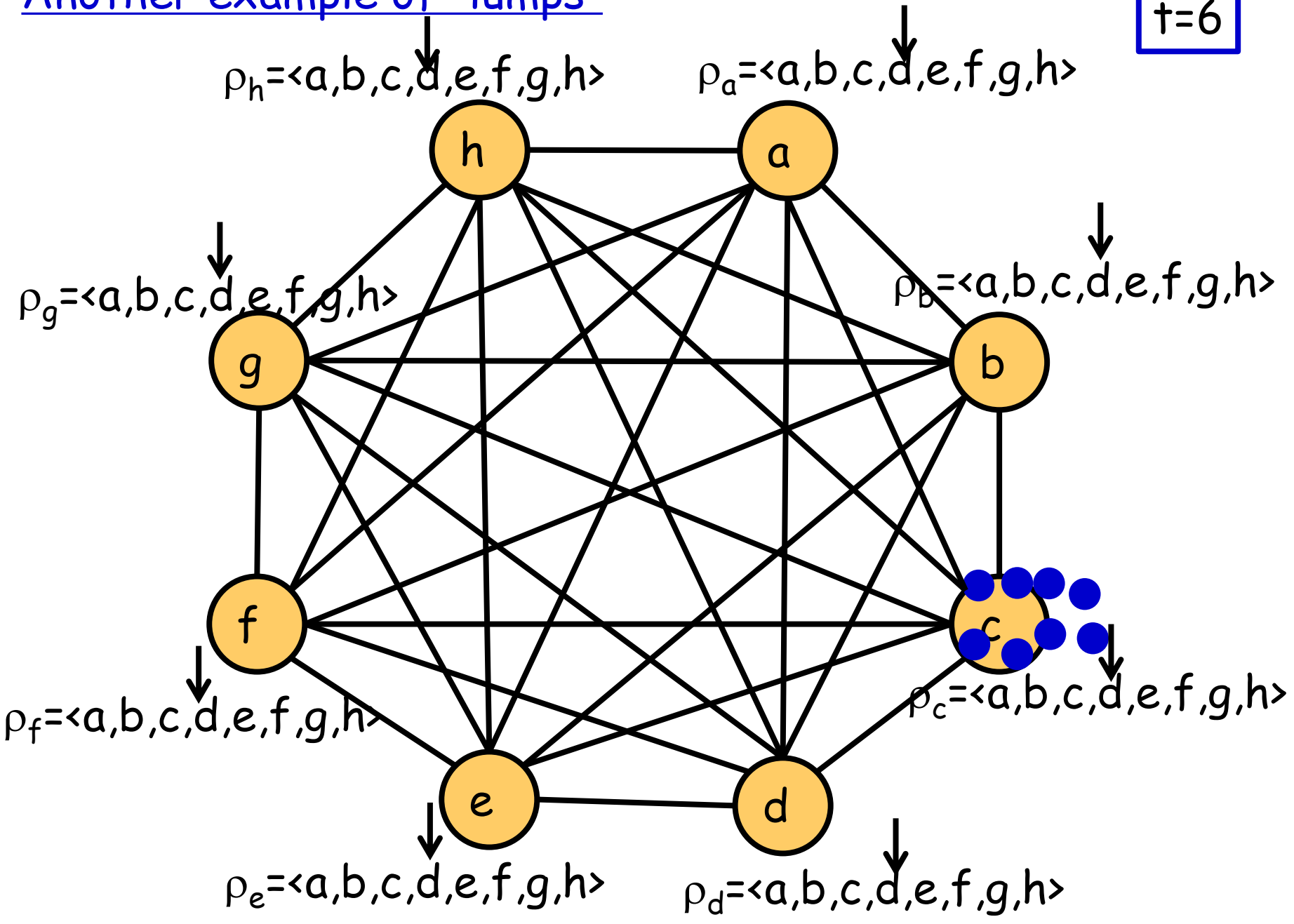
Another example of "lumps"

t=5



Another example of "lumps"

t=6



Main Results (1/3)

Thm. [K, Koga, Makino 10+]

There exists a multidigraph $\mathcal{G}$=(V,$\mathcal{E}$),

exists a ini. config, and exists a rotor-router, such that

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right| \geq \Omega(m)$$

holds, where m = $|\mathcal{E}|$,

$\chi_w^{(T)}$: #tokens @ w∈V, @ time T > 0 in det. RW

$\mu_w^{(T)}$: E[#tokens] @ w∈V , @ time T > 0 in RW

Main Results (2/3)

$\chi_w^{(T)}$: #tokens @ $w \in V$, @ time $T > 0$ in detRW

$\mu_w^{(T)}$: E[#tokens] @ $w \in V$, @ time $T > 0$ in RW

Thm. [K, Koga, Makino 10+]

If all eigenvalues of P are nonnegative, then

for any multidigraph, for any ini. config., for any rotor-router,

for any $w \in V$, for any time t,

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right| \leq (2m-n)\left(\max_{i\in\{1,\ldots,\kappa\}} n_i + n + 3\right) \leq 4mn + O(m)$$

holds where, $n=|V|$, $|E|=m$, n_i = size of i-th Jordan cell.

Remark

The c … "

- is s… chain,
  - ✓ which is usually us…
- When P is symmetric … efinite"

$$\left|\frac{\chi_w^{(T)}}{M} - \frac{\mu_w^{(T)}}{M}\right| = O\left(\frac{mn}{M}\right) \xrightarrow{M\to\infty} 0$$

remark

$\frac{\mu^{(T)}}{M} \simeq \pi^{(T)}$: distr. at time T

Main Results (3/3)

$\chi_w^{(T)}$: #tokens @ w∈ V , @ time T > 0 in detRW

$\mu_w^{(T)}$: E[#tokens] @ w∈ V , @ time T > 0 in RW

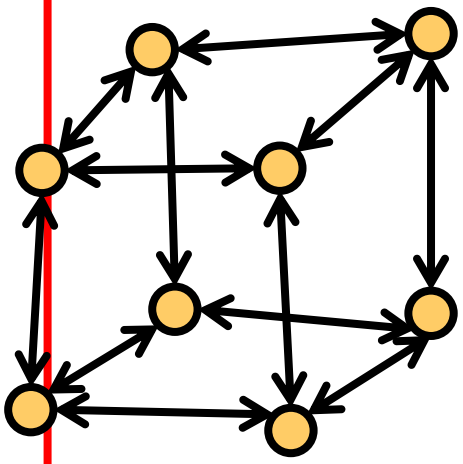
Thm. [K, Koga, Makino 10+]

On $\{0,1\}^d$ skeleton,

for any ini. config., for any rortor-router,

for any vertex w, for any time t,

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right| \leq \frac{3}{2}d^3 + \mathrm{O}(d^2)$$



Johnson graph J(d,c) =(V_J,E_J)

V_J={S ⊂ {1,…,d} | |S|=c},

E_J={ {S,T} ∈ V_J^2 | |S⊕T|=2 }

Thm. [K, Koga, Makino 10+]

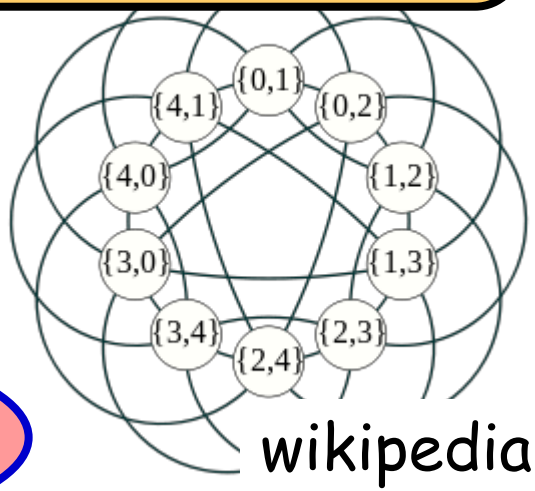
On Johnson graph J(d,c),

for any ini. config., for any rortor-router,

for any vertex w, for any time t,

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right| \leq 2c^3 \cdot (d-c)^2 + \mathrm{O}(c^3 \cdot (d-c))$$

poly log(#vertices)



wikipedia

ind. of #tokens M.

3.2. Previous works & our results

Thm. [Cooper & Spencer 2006]

On Z^d, there exists a constant C_d (depend. only d), such that for any ini. config., for any rotor-router, for any vertex w, for any time t,

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right| \leq C_d$$

Previous works about single vertex discrepancy(1/2)

2006	Cooper, Spencer	Rotor-router on Z^d ➢ v-disc. $\leq C_d$
2007	Cooper, Doerr, Spencer, Tardos	Rotor-router on Z^1 ➢ $C_1 \leq 2.29$
2008	Cooper, Doerr, Friedrich, Spencer	Rotor-router on infinite k-reg. tree ➢ v-disc. $> \Omega(\sqrt{kT})$ at time T
2009	Doerr, Friedrich	Rotor-router on Z^2 ➢ $C_2 \leq 7.83$ (rotor-router=$\uparrow\rightarrow\downarrow\leftarrow$) ➢ $C_2 \leq 7.29$ (rotor-router=$\uparrow\downarrow\leftarrow\rightarrow$)
2010+	K, Koga, Makino (this work)	Rotor-router on finite multidigraph ➢ v-disc. $\leq 4mn+O(m)$ Rotor-router on $\{0,1\}^d$ ➢ v-disc. $\leq O(d^3)$ (poly log(#vertices))
2011+	Kajino, K., Makino	Next page
2012+	Shiraga et al.	Next page

Previous works about single vertex discrepancy(2/2)

2012 (2010)	Kijima, Koga, Makino	Rotor-router on multidigraph ➢ v-disc. $\leq 4m^*n+O(m^*)$ (P: rational + ergodic + reversible + lazy) Rotor-router on $\{0,1\}^d$ ➢ v-disc. $\leq O(d^3)$ (poly log of #vertices)
2012+	Kajino, Kijima, Makino	Rotor-router on multidigraph ➢ v-disc. $\leq O\left(\alpha \cdot \frac{m^*n}{1-\lambda}\right)$ (P: rational + irreducible) Rotor-router on $\{0,1\}^d$ ➢ v-disc. $\leq O(d^2)$ (poly log of #vertices)
2012+	Shiraga, Yamauchi, Kijima, Yamashita	Functional-router on (simple)digraph ➢ v-disc. $\leq O\left(\sqrt{\frac{\pi_{\max}}{\pi_{\min}}} \cdot \frac{mn}{1-\lambda} \cdot \log M\right)$ (P: real + ergodic + reversible)

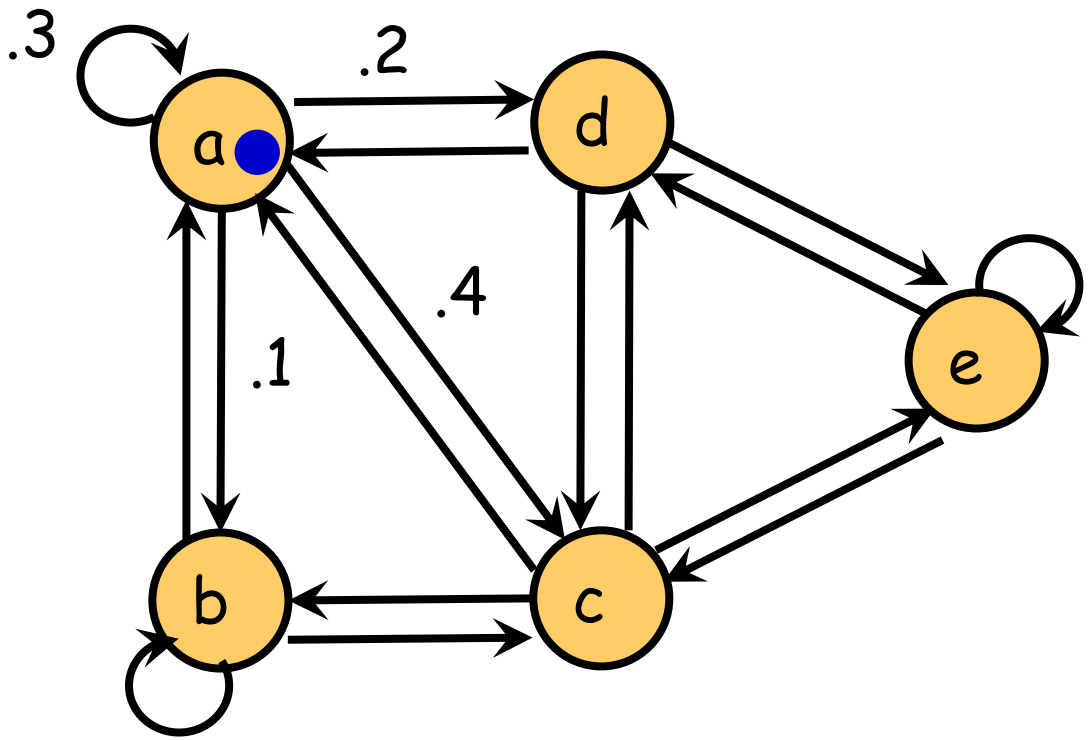
3.3. Functional-router model

We propose a new deterministic process
which imitates irrational transition probabilities,
in a similar fashion to the rotor-router model.

In Random Walk

A token randomly walks on a graph.

- ✓ π^0: ini. distr. (token is on v at time 0, w.p. $(\pi^0)_v$)
- ✓ P: trans. prob. matrix (move u-> v w.p. P_{uv})
- ✓ distr. $\pi^t := \pi^0 P^t$ at time t (token is on v at time t w.p. $(\pi^t)_v$)



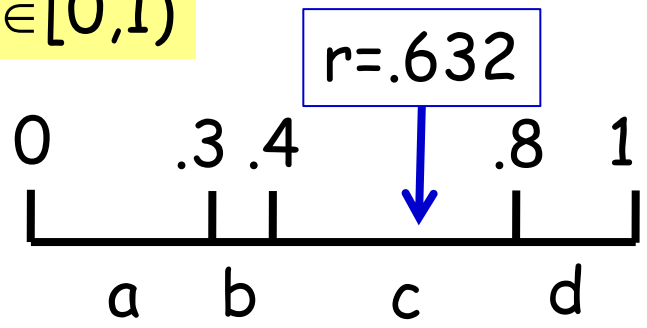
In Random Walk

A token randomly walks on a graph.

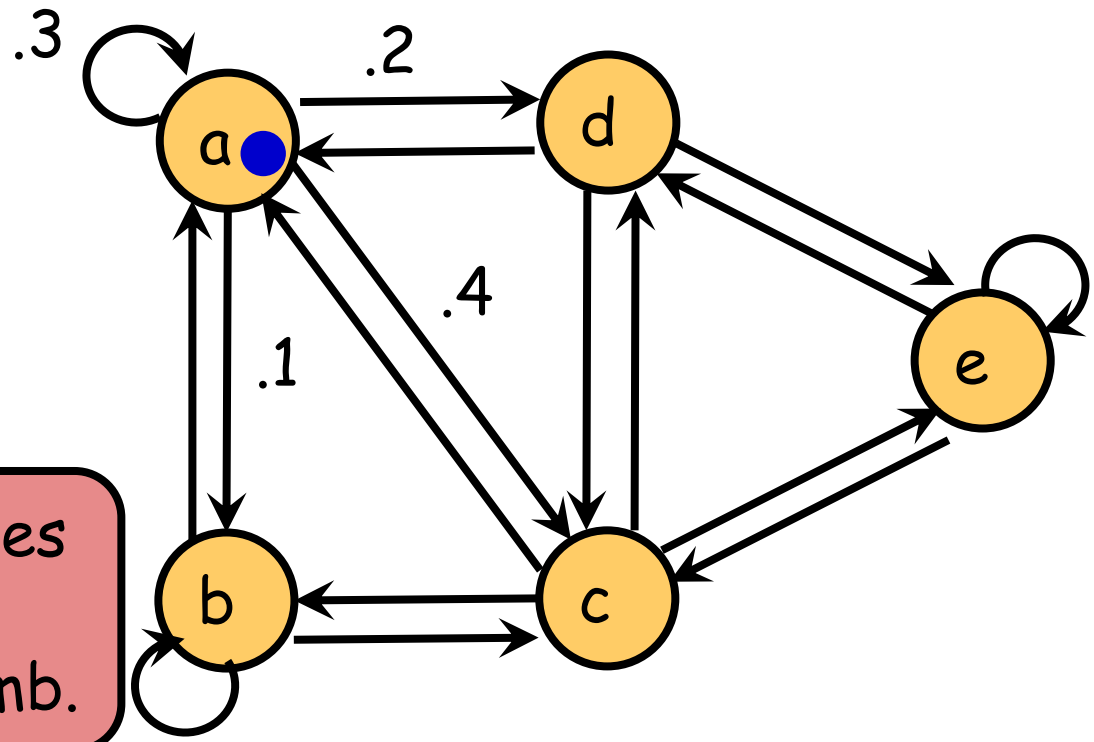
- ✓ π^0: ini. distr. (token is on v at time 0, w.p. $(\pi^0)_v$)
- ✓ P: trans. prob. matrix (move u-> v w.p. P_{uv})
- ✓ distr. $\pi^t := \pi^0 P^t$ at time t (token is on v at time t w.p. $(\pi^t)_v$)

A RW is implemented using random number.

$r \in [0,1)$



Func.-router model uses low discrepancy seq. instead of random numb.



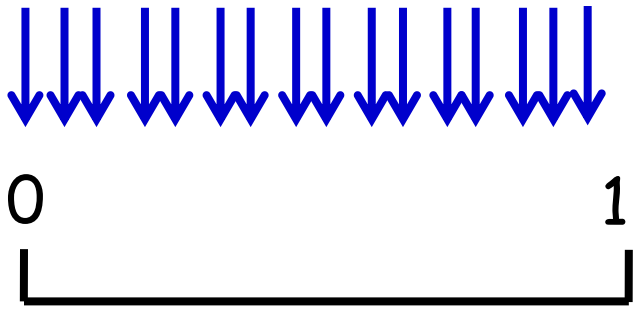
Van der Corput seq. --- a low discrepancy sequences

For an integer $i = \sum_{j=0}^{\lfloor \lg i \rfloor} \beta_j(i) 2^j$

where $\beta_j(i) \in \{0,1\}$ $(j = 0,1, \dots, \lfloor \lg i \rfloor)$

$$\psi(i) := \sum_{j=0}^{\lfloor \lg i \rfloor} \beta_j(i) 2^{-(j+1)}$$

i	$(i)_2$	$(\psi(i))_2$	$\psi(i)$
0	0	0	0
1	1	0.1	1/2
2	10	0.01	1/4
3	11	0.11	3/4
4	100	0.001	1/8
5	101	0.101	5/8
6	110	0.011	3/8
…	…	…	…

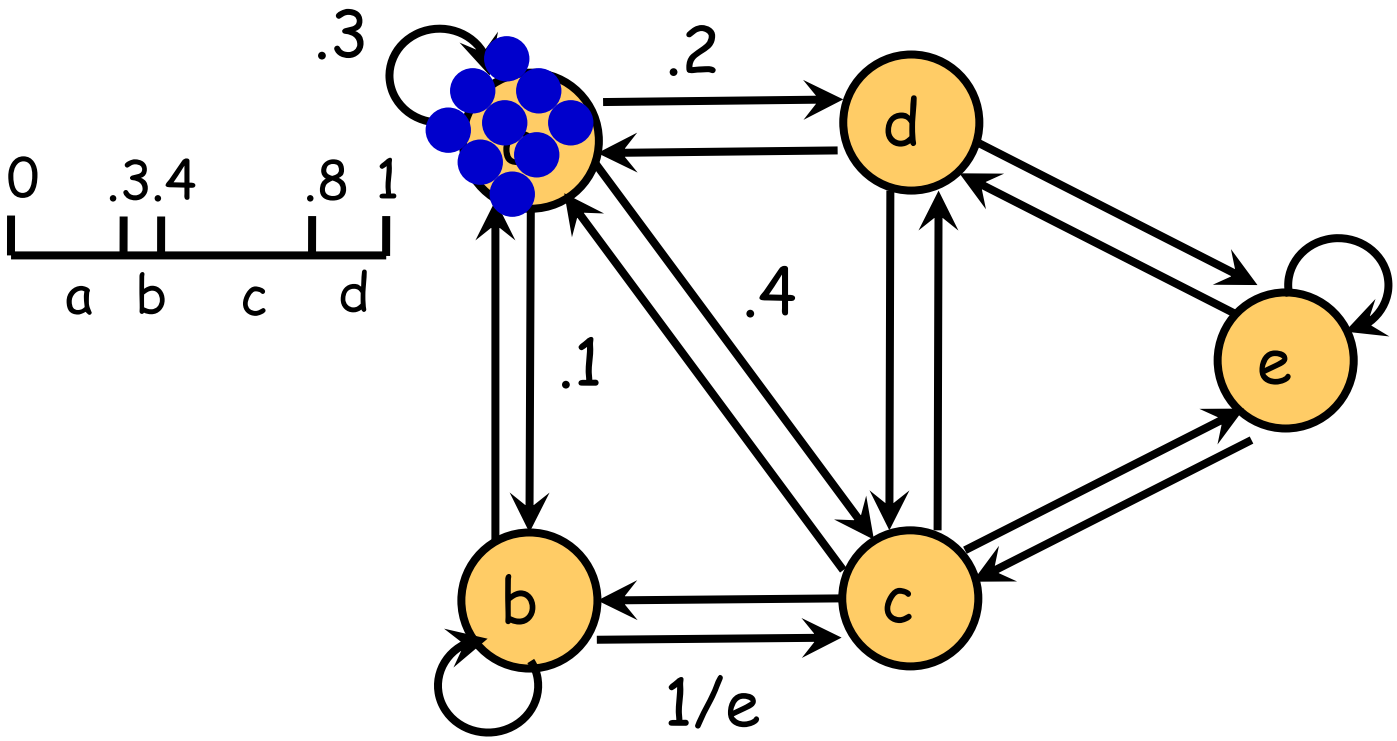


Functional-router model --in a similar fashion to rotor-router

M tokens deterministically walks on a graph, using low disc. seq.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

t=0

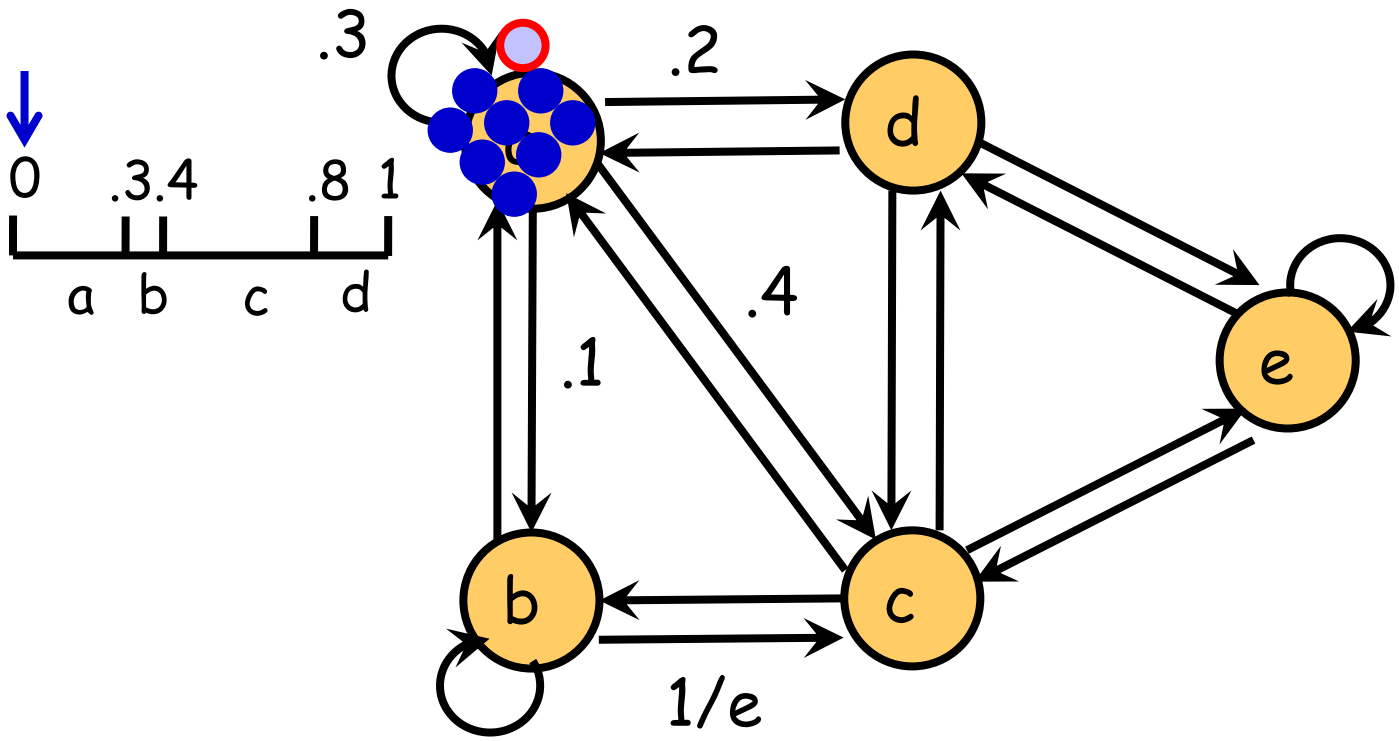


In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

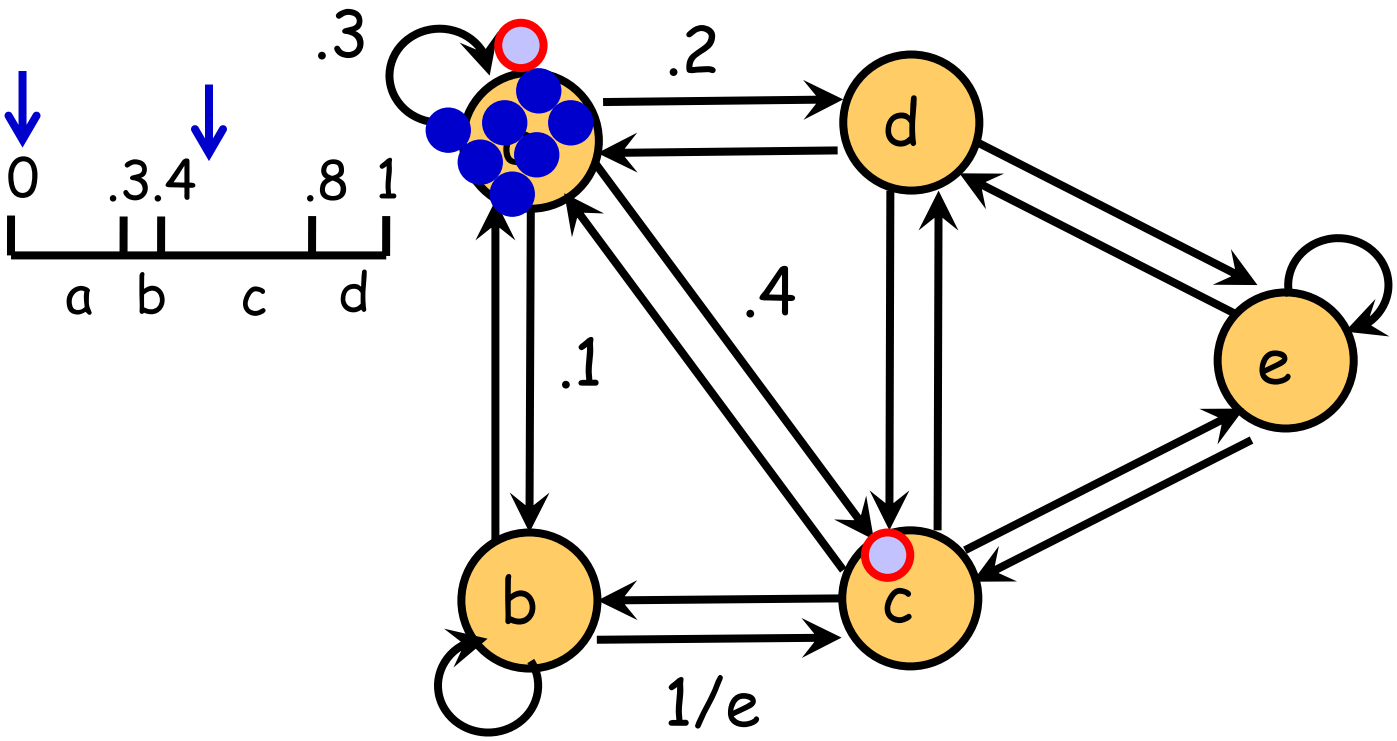


In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

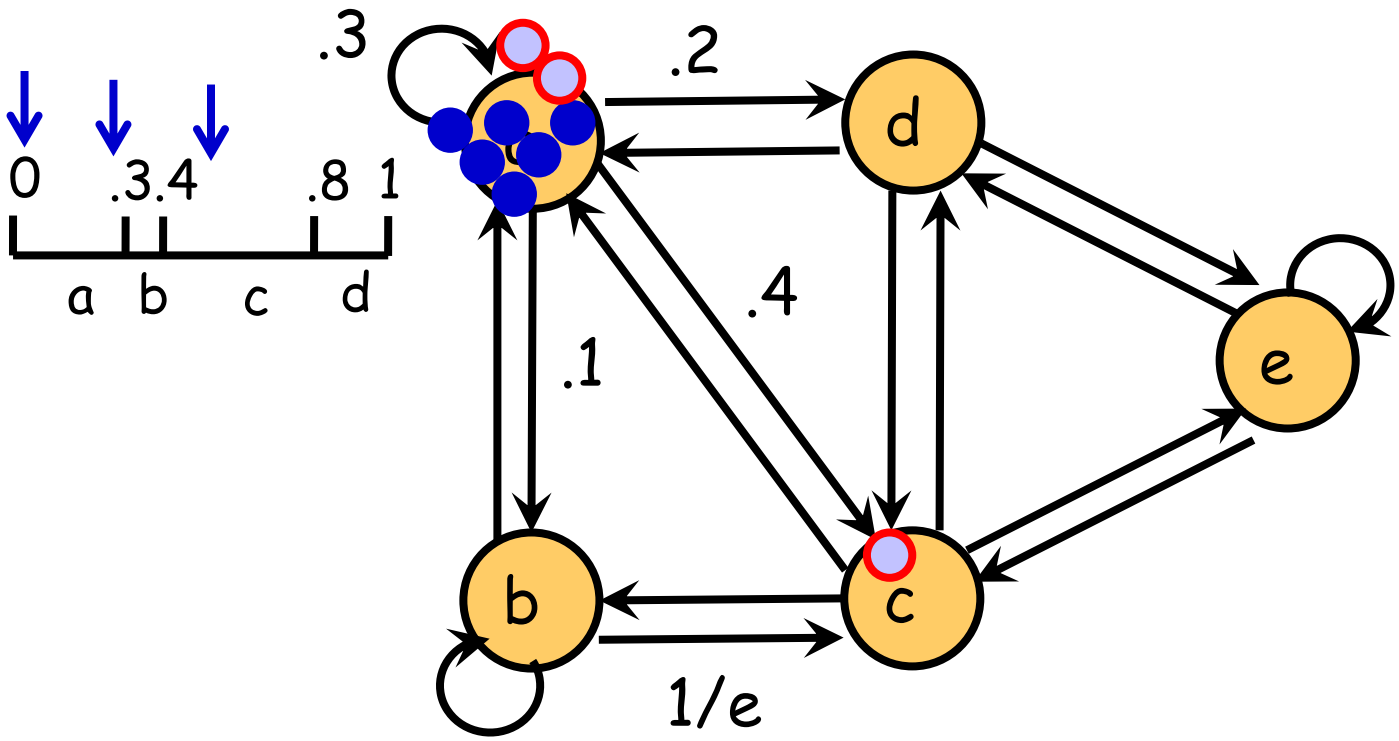


In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

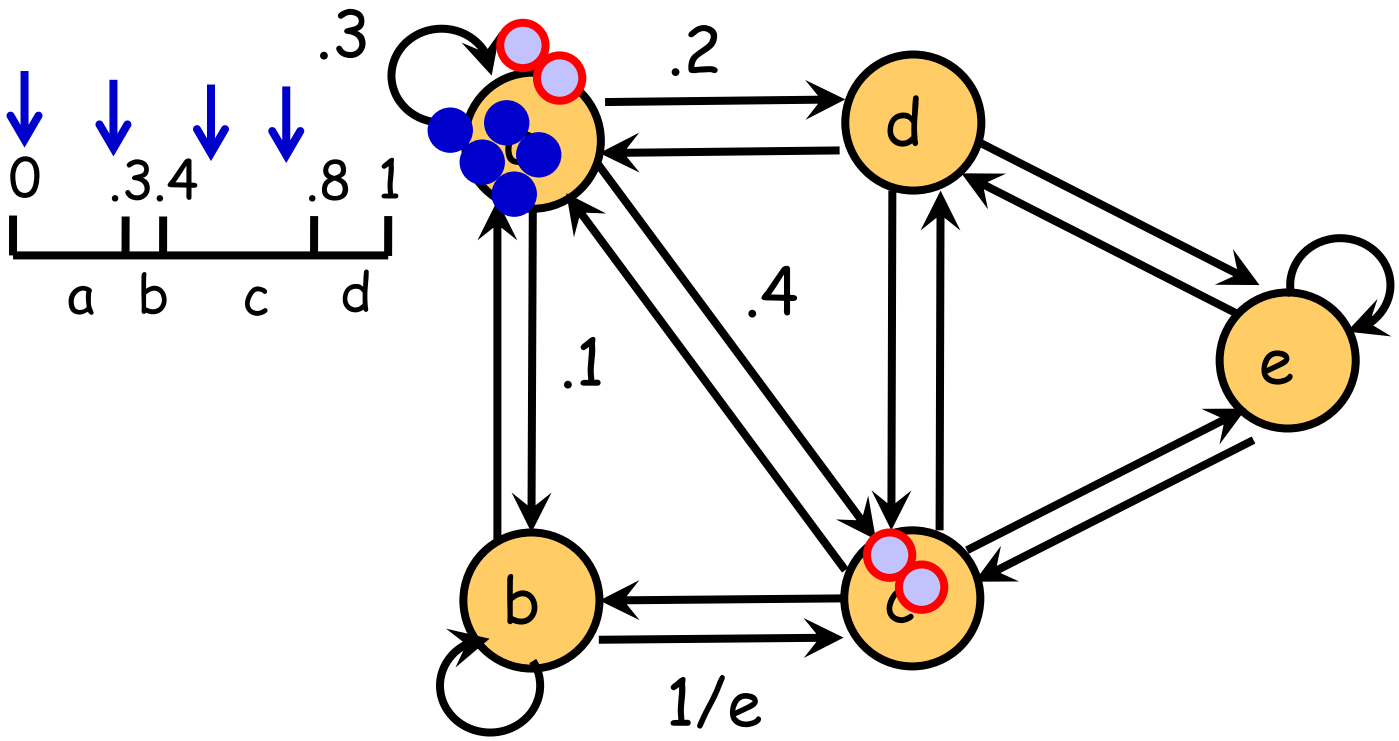


In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

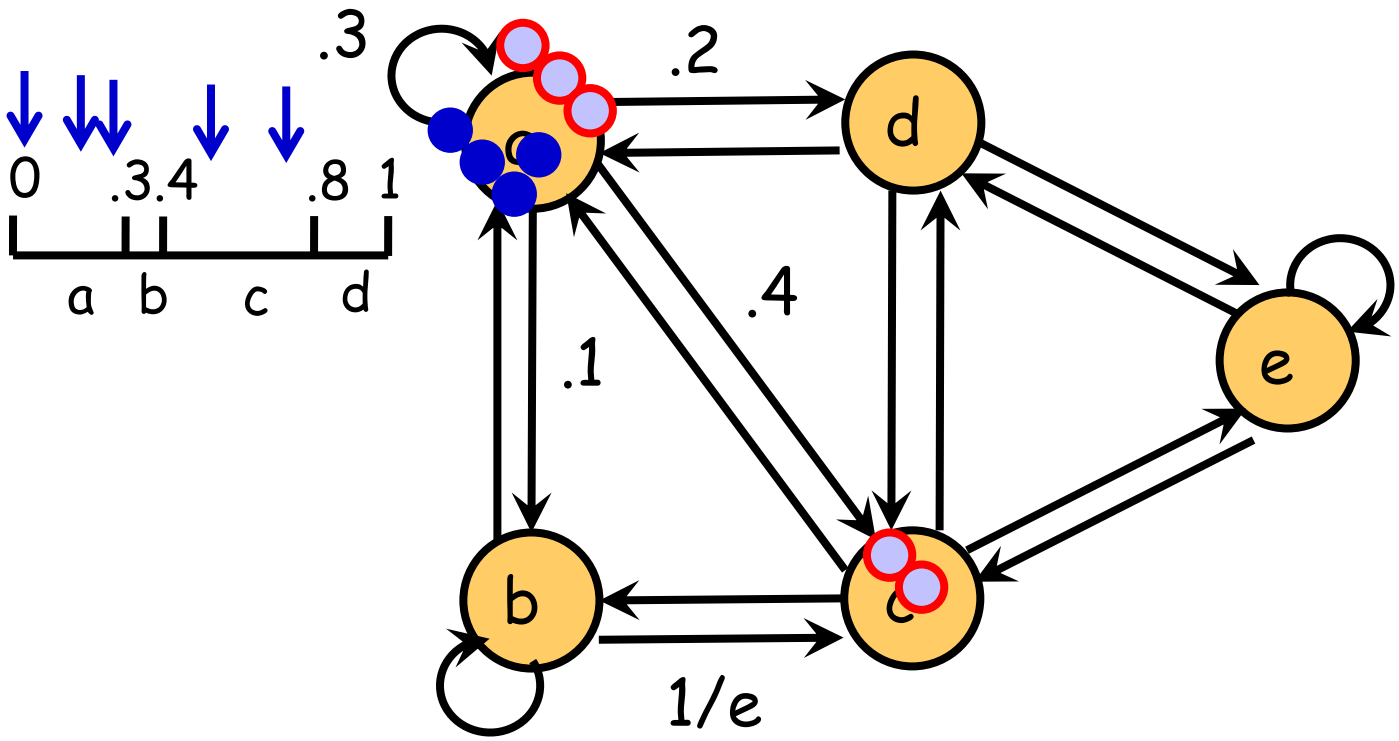


In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

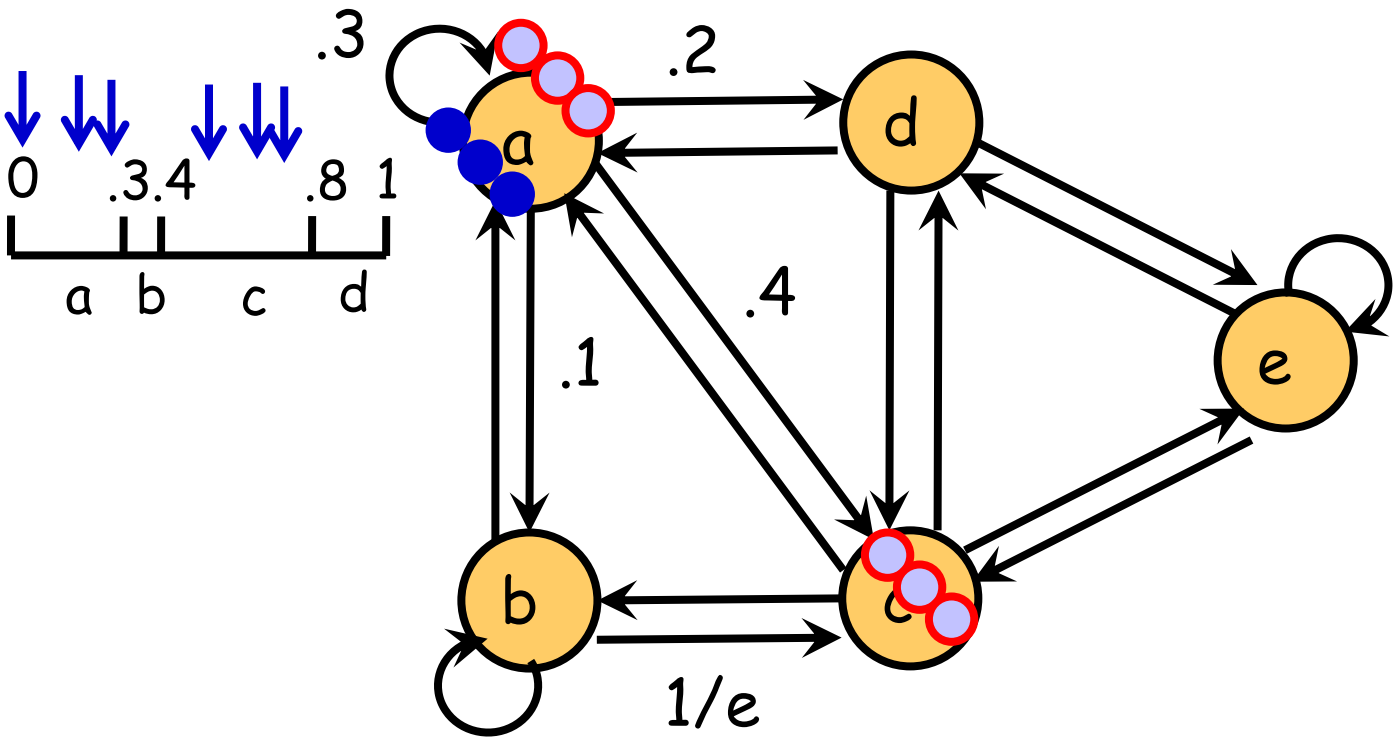


In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

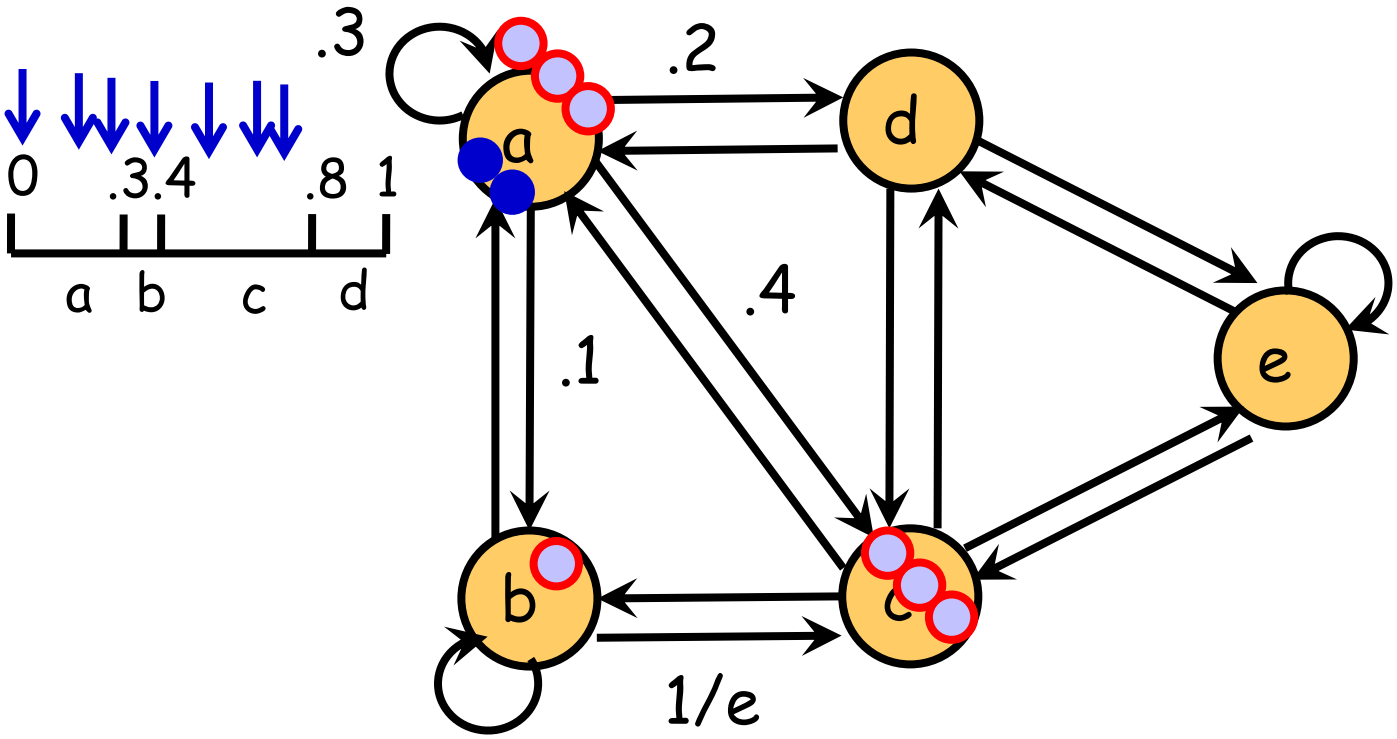


In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

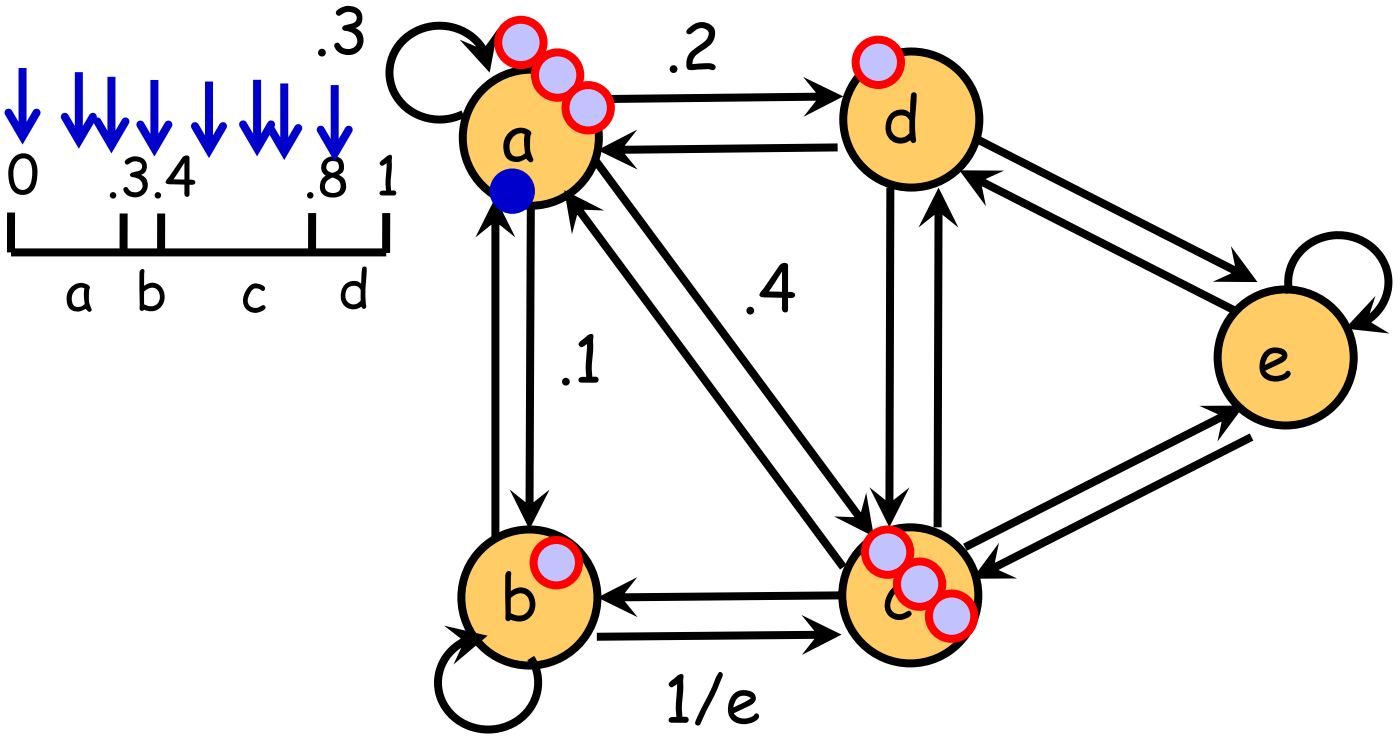


In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

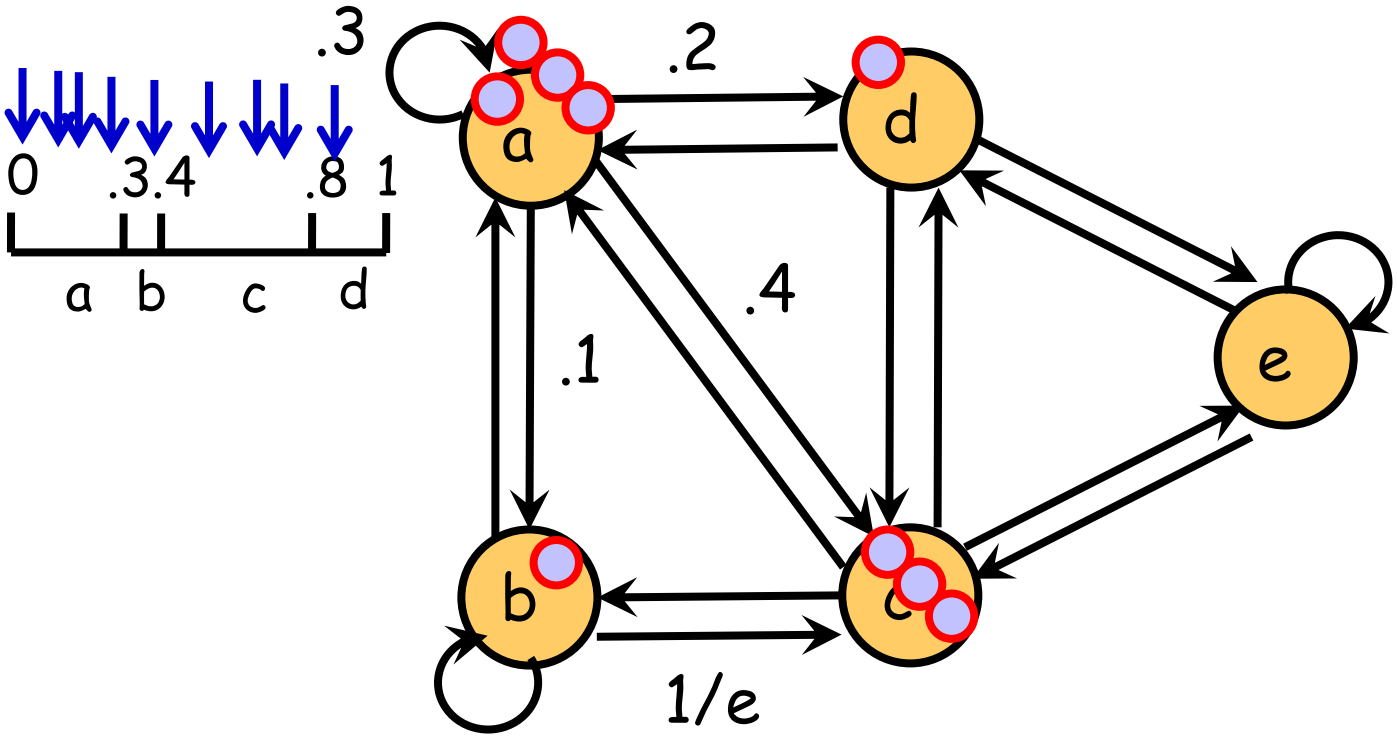


In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

$t \in (0,1)$

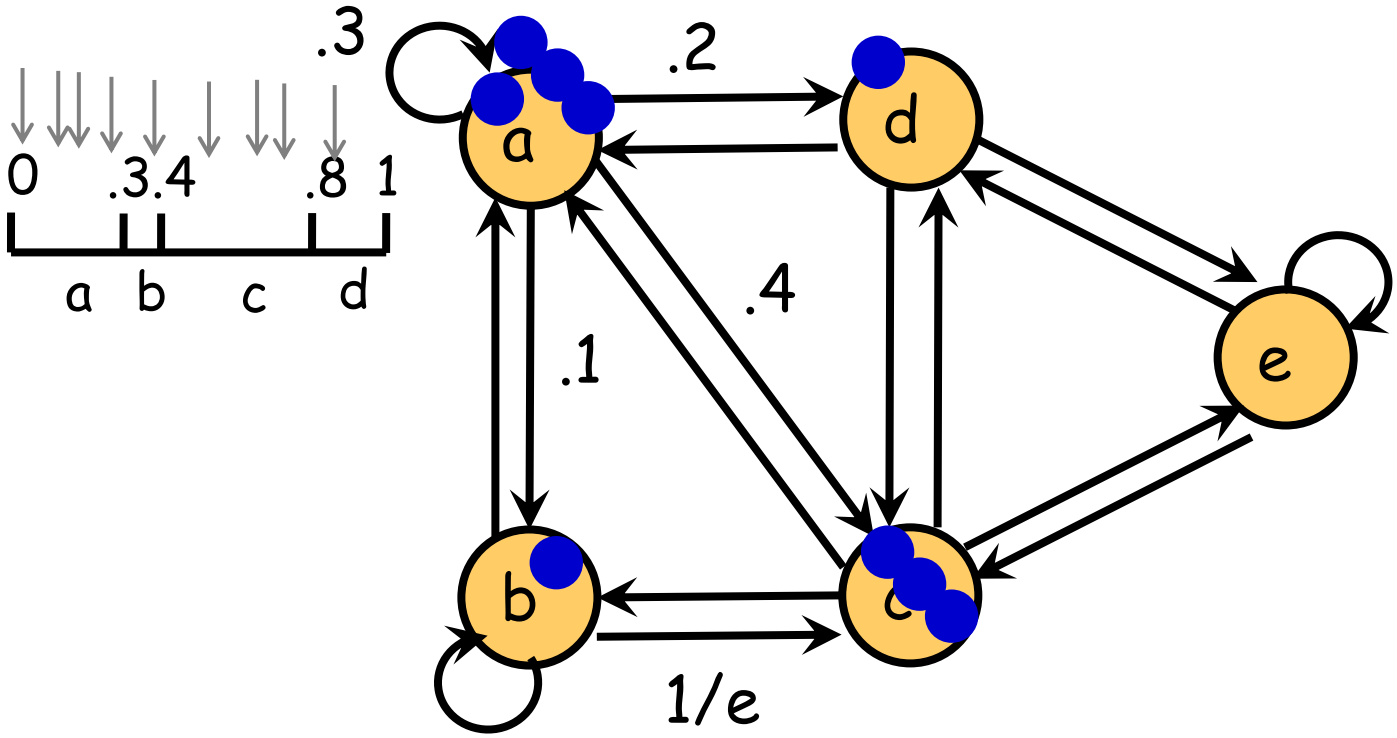


In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

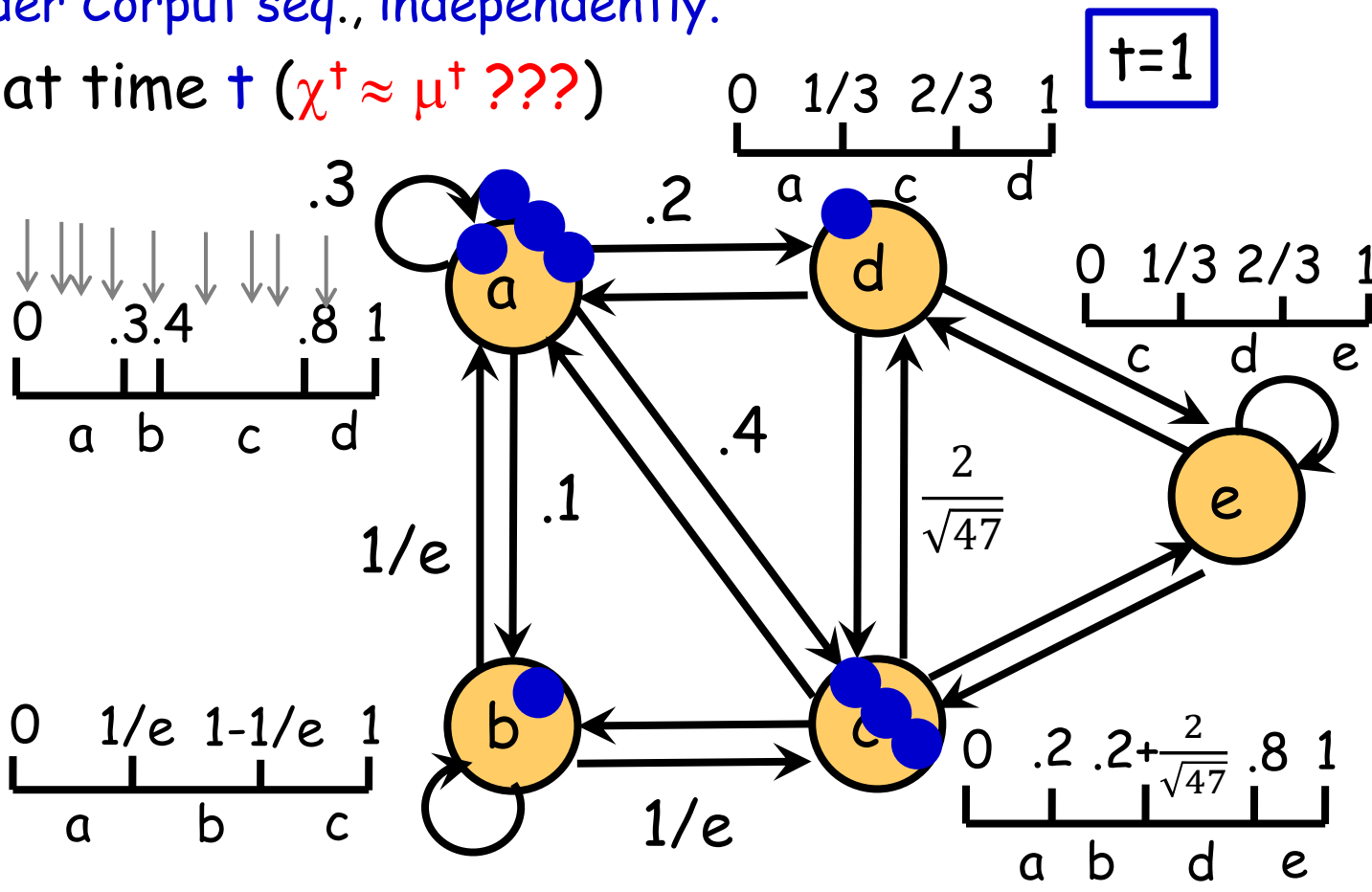
t=1



In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

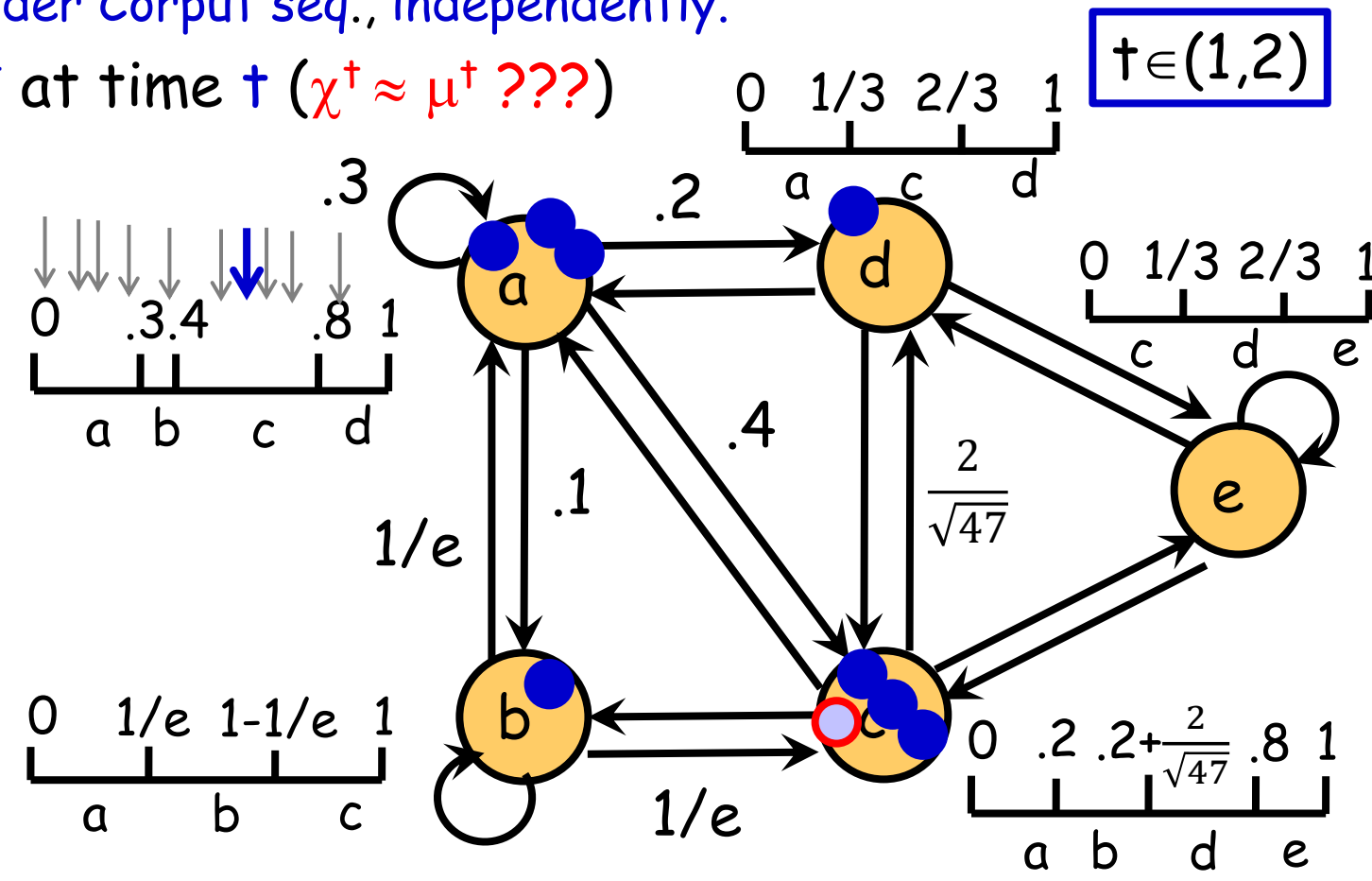
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

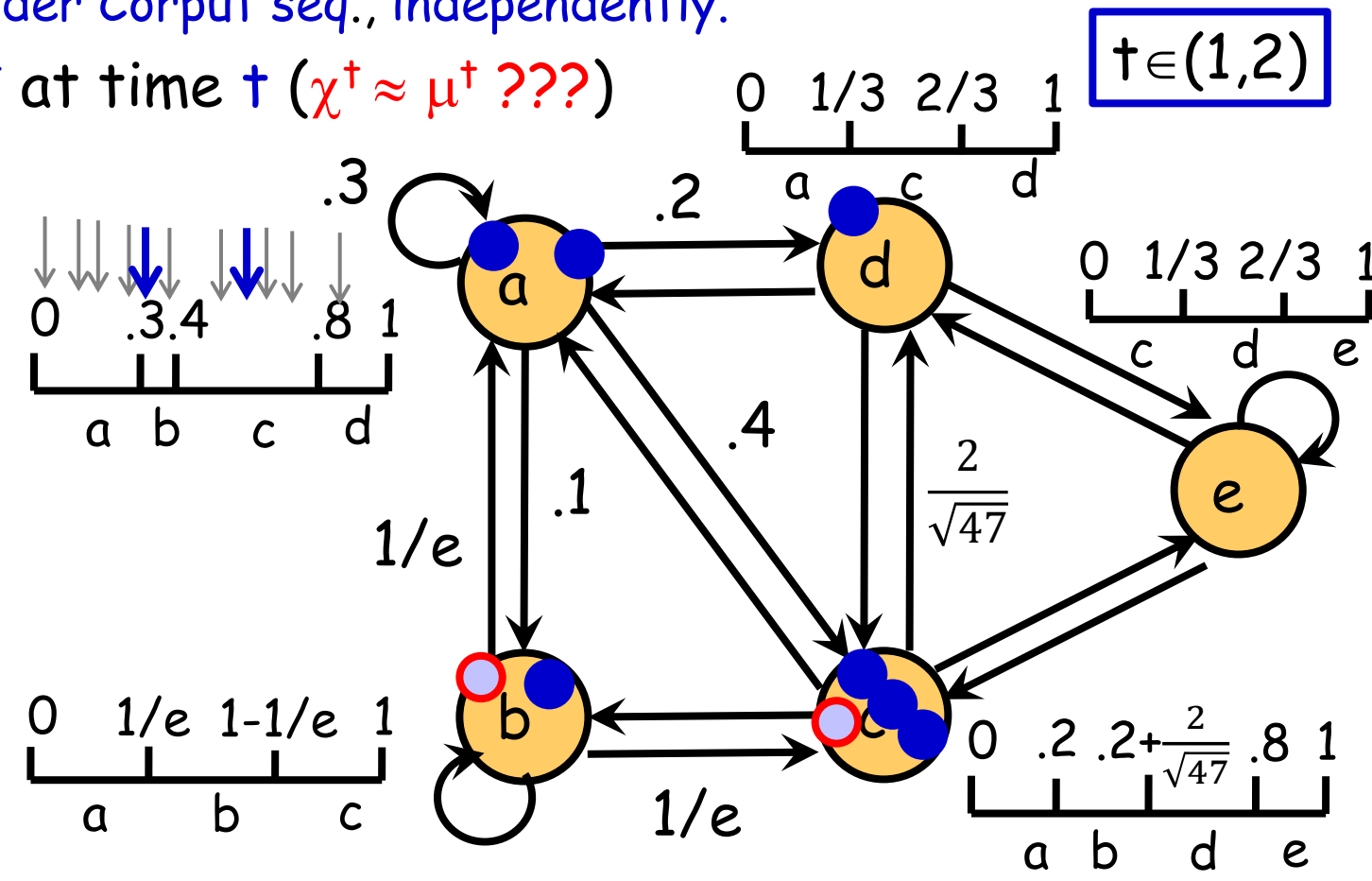
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

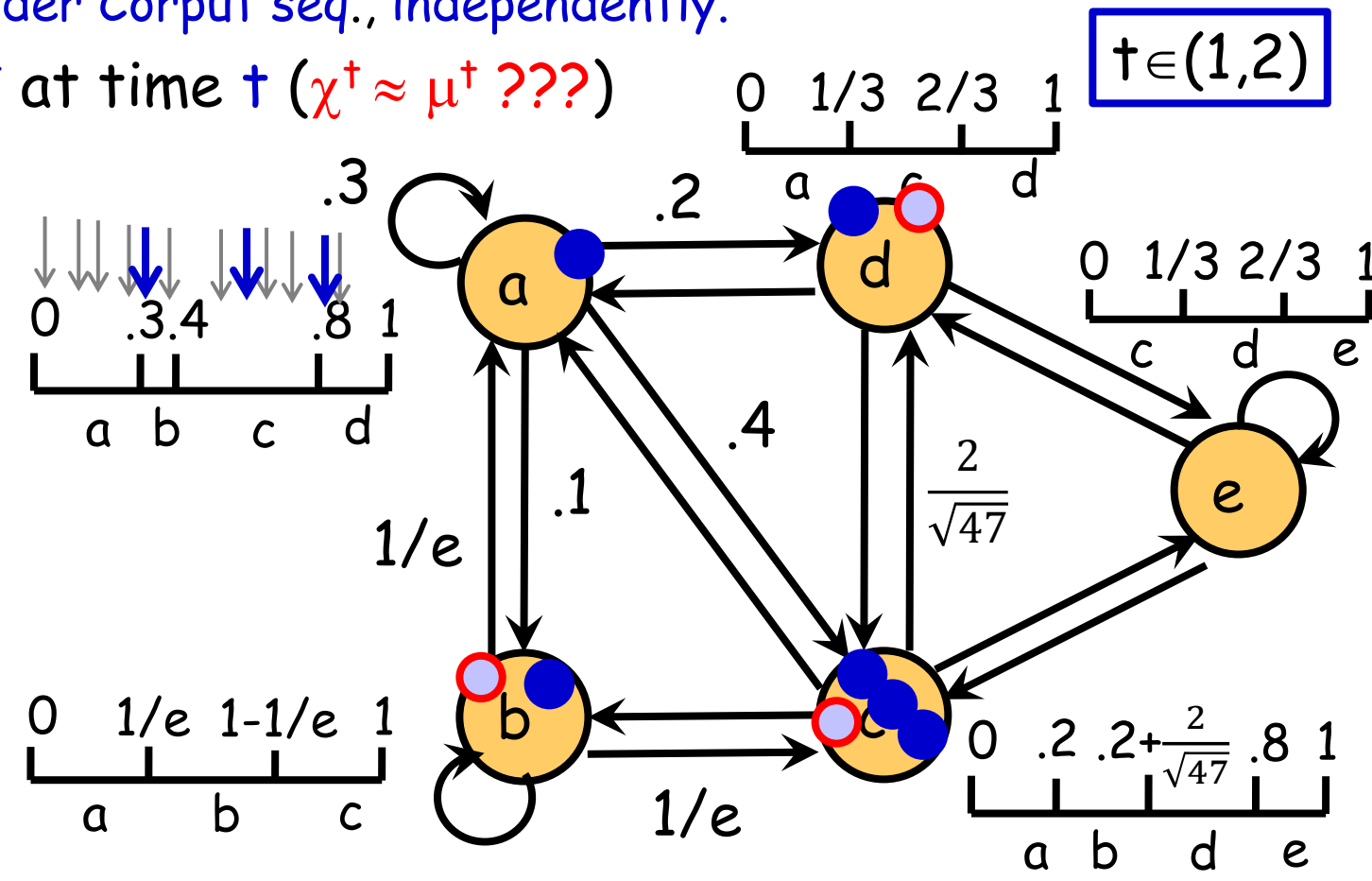
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

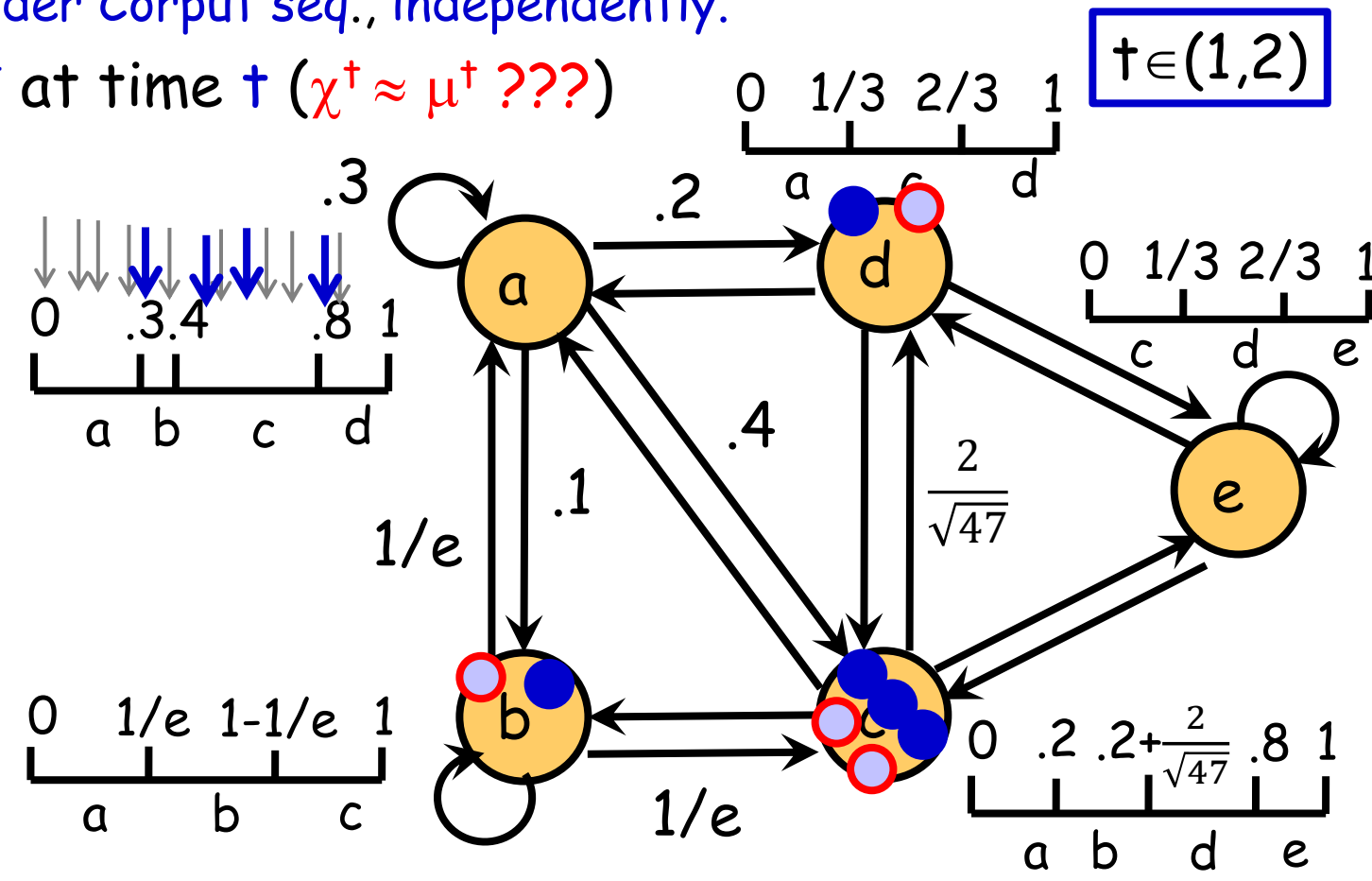
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

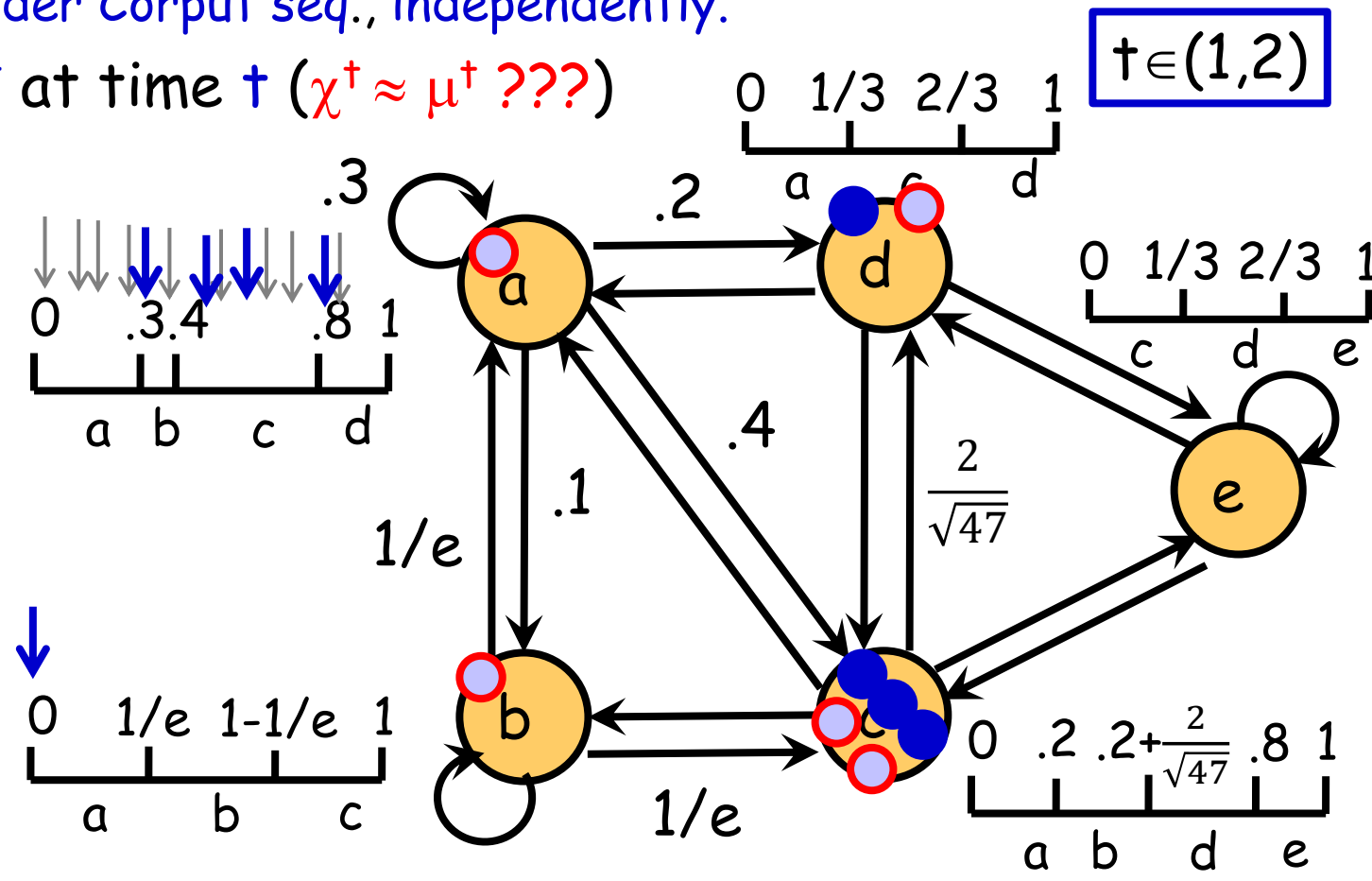
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

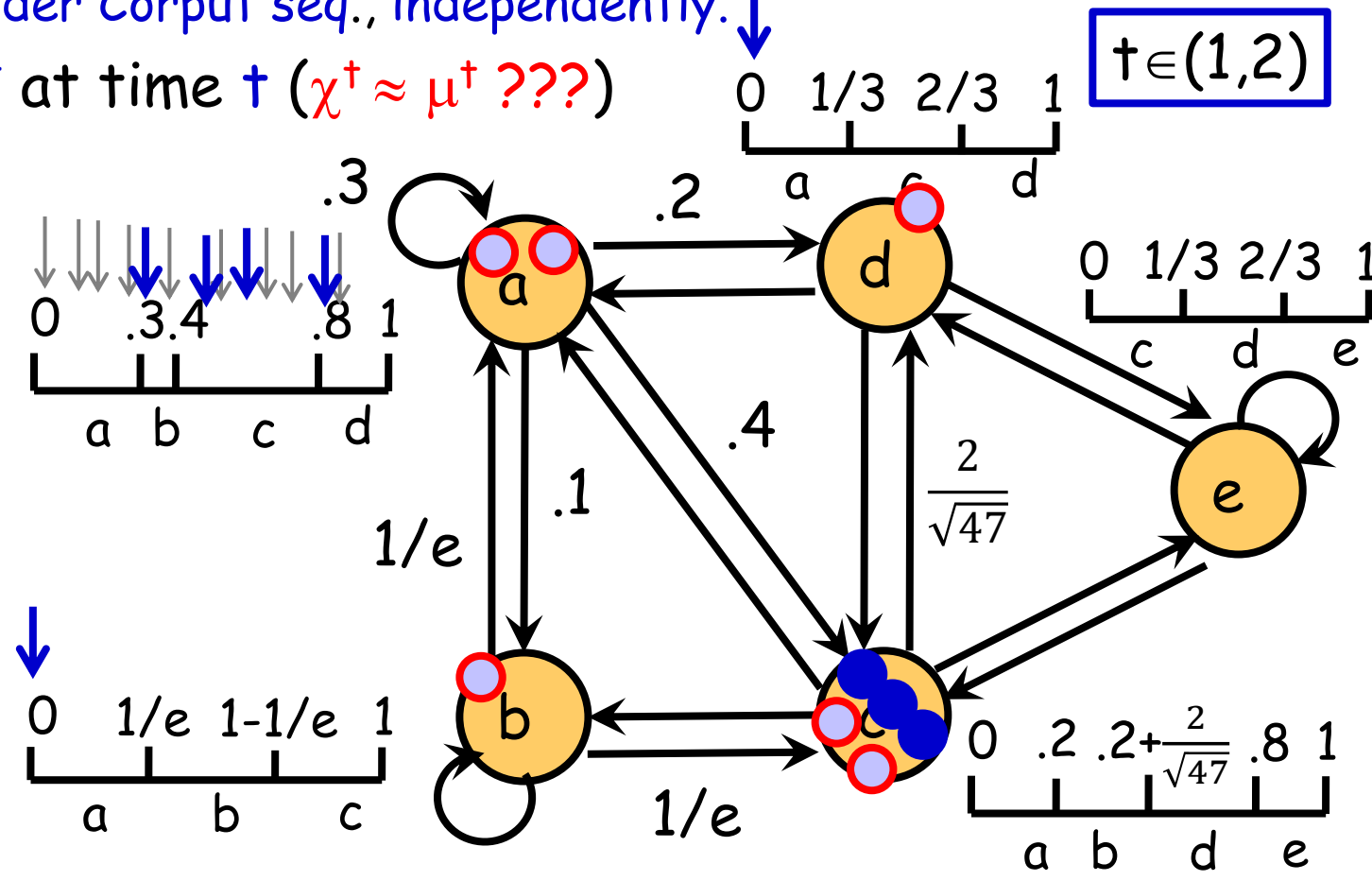
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv}
  using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

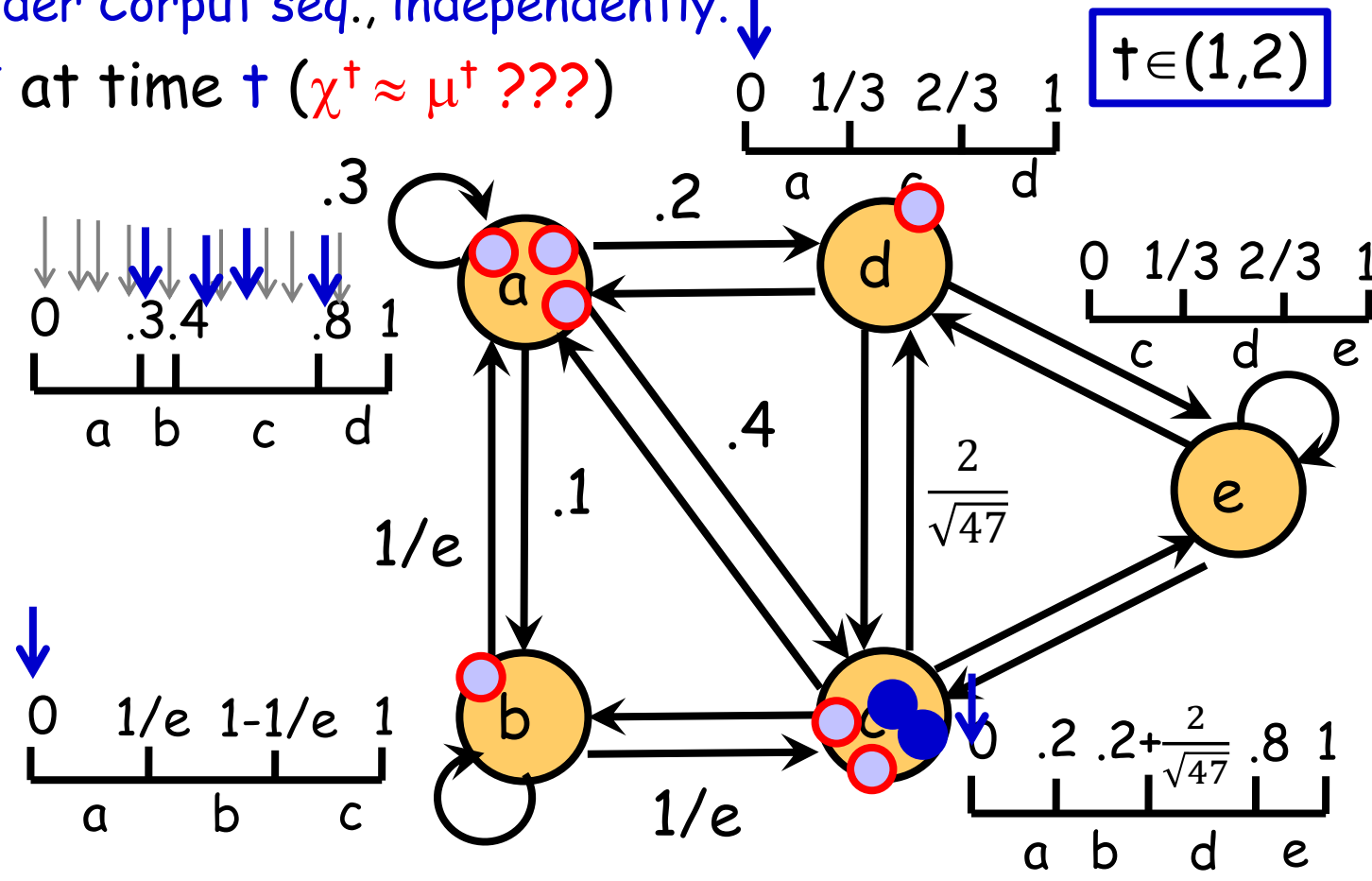
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv} using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

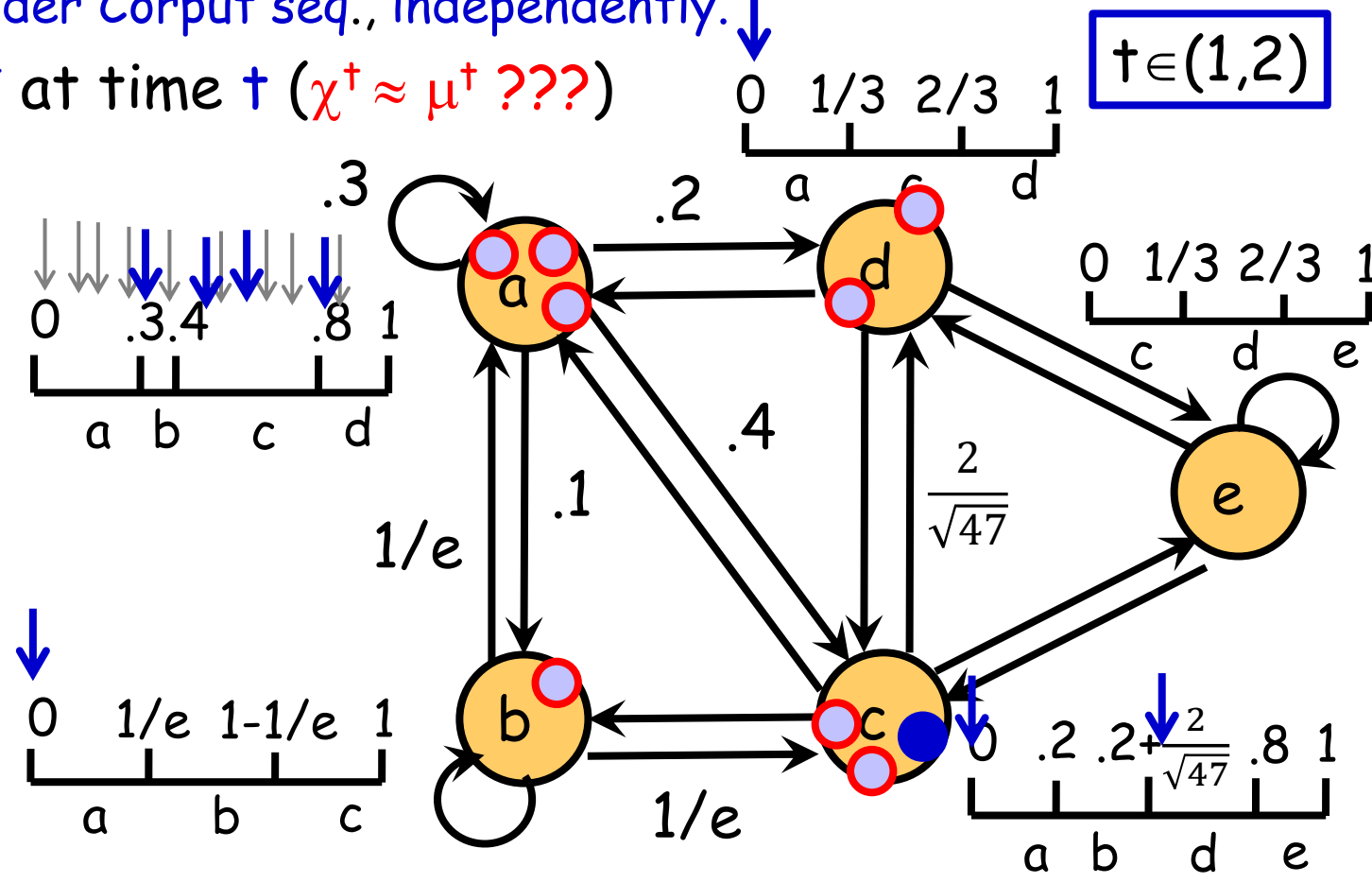
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv} using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

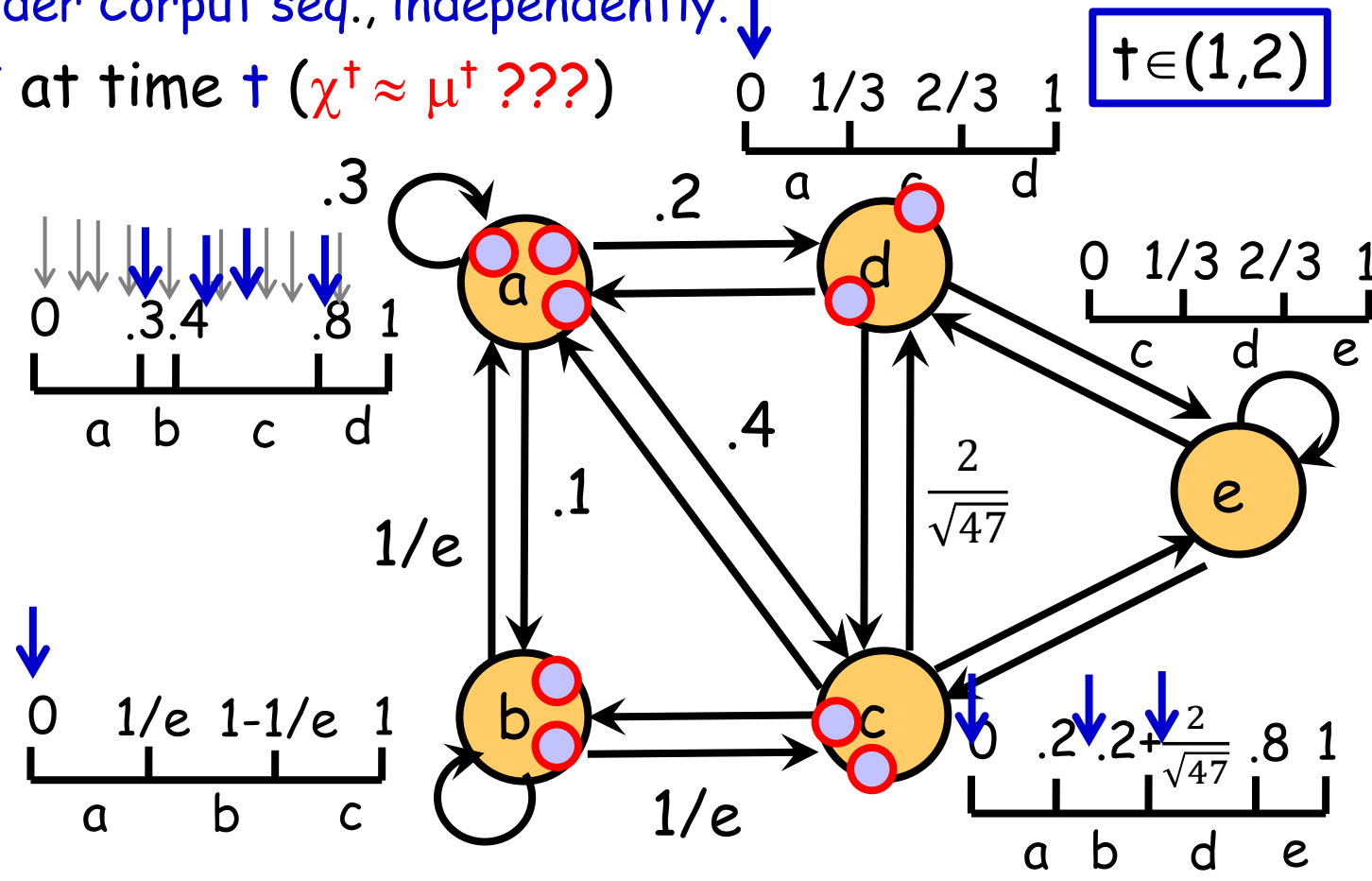
- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv} using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)



In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv} using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

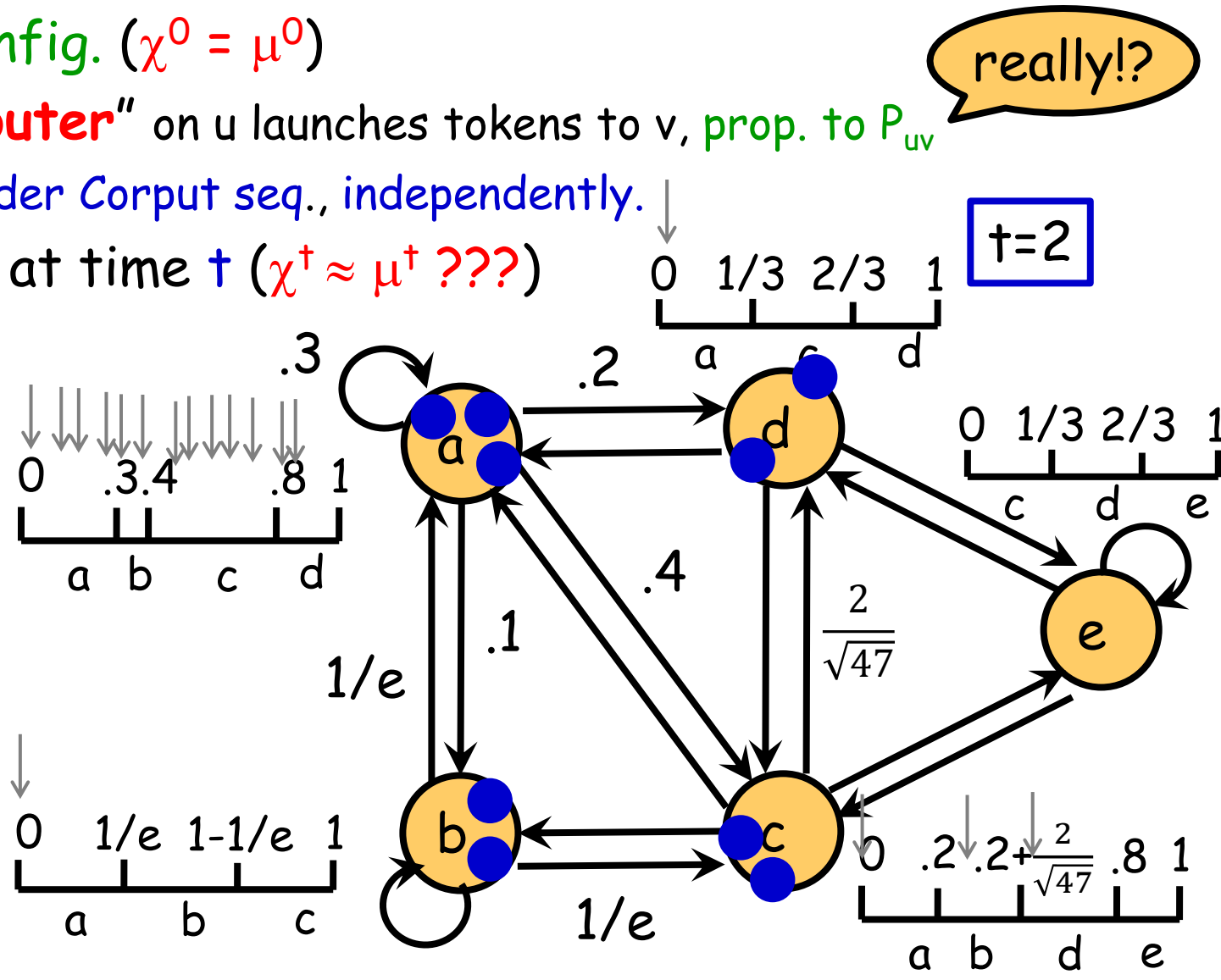


In Functional-router model

M tokens deterministically walks on a graph, using low disc. seq.

- ✓ χ^0: ini. config. ($\chi^0 = \mu^0$)
- ✓ "**func.-router**" on u launches tokens to v, prop. to P_{uv} using Van der Corput seq., independently.
- ✓ config. χ^t at time t ($\chi^t \approx \mu^t$???)

really!?



The disc. of Van der Corput seq.

$|I_{u,v}[0,z)|$: #tokens reaching at v out of z tokens launched from u, in total.

Thm. [Shiraga et al. 12+]

For any transition matrix P,

$$\left| \frac{|I_{u,v}[0,z)|}{z} - P(u,v) \right| < \frac{2\lfloor \lg z \rfloor + 2}{z} = O\left(\frac{\log z}{z}\right)$$

holds for any $u, v \in V$ and any $z \in \mathbb{Z}_{z \geq 0}$.

Unfortunately this bound is tight.

Prop. [Shiraga et al. 12+]

Exists a transition matrix P,

$$\left| \frac{|I_{u,v}[0,z)|}{z} - P(u,v) \right| > \frac{\lg\left(\frac{3}{4}z\right)}{3z} = \Omega\left(\frac{\log z}{z}\right)$$

holds for any $u, v \in V$ and any $z \in \mathbb{Z}_{z \geq 0}$.

Example for the lower bound

Prop. [Shiraga, Yamauchi, K., Yamashita 12+]

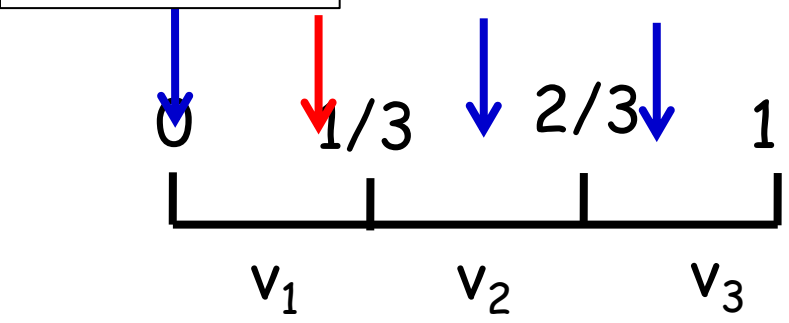
Exists an example such that $\left|\chi_w^{(T)} - \mu_w^{(T)}\right| > \lg\left(\frac{3}{4}M\right)$ holds, for infinitely many $M \in \mathbb{Z}_{>0}$: #tokens.

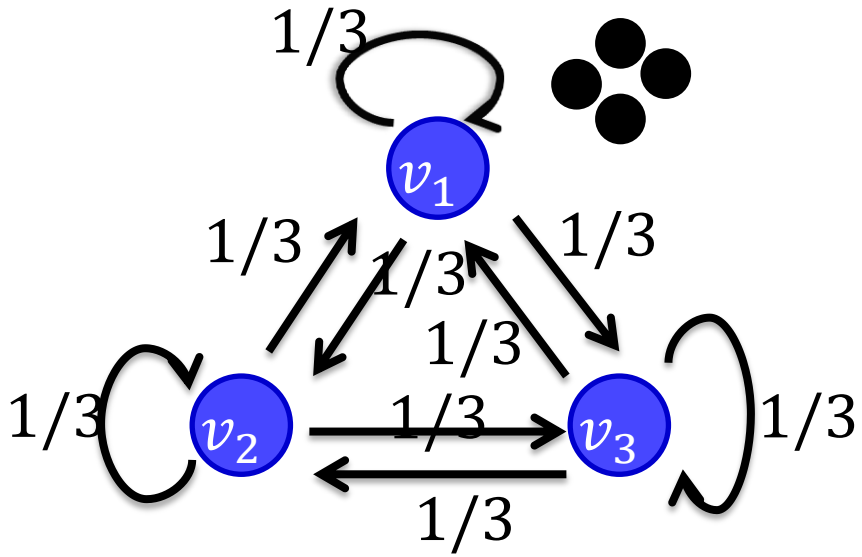
Consider a simple random walk on K_3.

Let $M := \sum_{l=1}^{k} 4^l$ be the number of tokens (for any $k \in \mathbb{Z}_{>0}$).

Note that $4^l \equiv 1 \pmod 3$
since $4^l - 1 = 3(4^{l-1} + \cdots + 1)$

intuitively





Main Thm.

Thm. [Shiraga, Yamauchi, K., Yamashita 12+]

If P is ergodic and reversible, then

for any ini. config., for any $w \in V$, for any time t,

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right| < \sqrt{\frac{\pi(w)}{\pi_{\min}}} \cdot \frac{m(n-1)}{1-\lambda^*} \cdot 2(\lg M + 1)$$

holds where, $n=|V|$, $m=|E|$,

λ^* is the second largest eigenvalue of P,

π is the st

P is

$\pi(u)$ P(u,

holds for any $u,v \in V$.

$$\left|\frac{\chi_w^{(T)}}{M} - \frac{\mu_w^{(T)}}{M}\right| = O\left(\frac{\log M}{M}\right) \xrightarrow{M\to\infty} 0$$

MC

remark

$\frac{\mu^{(T)}}{M} \simeq \pi^{(T)}$: distr. at time T

A sketch of proof (1/4)

$\chi_w^{(T)}$: #tokens @ w∈ V , @ time T > 0 in detRW

$\mu_w^{(T)}$: E[#tokens] @ w∈ V , @ time T > 0 in RW

$Z_{v,u}^{(t)}$: #tokens v→u (v,u∈V) during [t,t+1)

Rem. $\sum_{u\in N(v)} Z_{v,u}^{(t)} = \chi_v^{(t)}$

understanding [Cooper, Spencer2006]

by transition matrix P

Lemma 1.

$$\chi_w^{(T)} - \mu_w^{(T)} = \sum_{t=0}^{T-1} \sum_{v\in V} \sum_{u\in N(v)} \left(Z_{v,u}^{(t)} - \chi_v^{(t)} P(v,u) \right) P^{T-t-1}(u,w)$$

Proof sketch.

$$\chi^{(T)} - \mu^{(T)} = \chi^{(T)} - \mu^{(0)} P^T = \chi^{(T)} P^0 - \chi^{(0)} P^T$$

$$= \sum_{t=0}^{T-1} \left(\chi^{(t+1)} P^{T-t-1} - \chi^{(t)} P^{T-t} \right)$$

$$= \sum_{t=0}^{T-1} \left(\chi^{(t+1)} - \chi^{(t)} P \right) P^{T-t-1}$$

Carefully considering each element, we obtain the claim.

■

A sketch of proof (2/4)

It is known that

$\Pi^{1/2}$ P $\Pi^{-1/2}$ is symmetric if P is reversible,

where $\Pi^{1/2} = \mathrm{diag}(\sqrt{\pi(1)}, \ldots, \sqrt{\pi(n)})$.

Thus, P is diagonalized as $\Lambda = B^{\top} \Pi^{1/2} P \Pi^{-1/2} B$,

where B is orthonormal, and each eigenvalue is real.

Then,

$$P^z(v,u) = \boldsymbol{e_v} P^z \boldsymbol{e_u} = \boldsymbol{e_v} B^{\top} \Pi^{-\frac{1}{2}} \Lambda^z \Pi^{\frac{1}{2}} B \boldsymbol{e_u} = \frac{1}{\sqrt{\pi(v)}} e_v B^{\top} \Lambda^z B \sqrt{\pi(u)} \boldsymbol{e_u}$$

$$= \sqrt{\frac{\pi(u)}{\pi(v)}} \sum_{j=1}^{n} e_v b_j^{\top} (\lambda_j)^z b_j e_u^{\top} = \sqrt{\frac{\pi(u)}{\pi(v)}} \sum_{j=1}^{n} (\lambda_j)^z b_j(v) b_j(u)$$

A sketch of proof (3/4)

Lemma 2.

$$\sum_{u \in N(v)} \left(Z_{v,u}^{(t)} - \chi_v^{(t)} P(v,u) \right) \sqrt{\frac{\pi(w)}{\pi(u)}} b_1(v) b_1(u) = 0$$

since

$$b_1^{\mathsf{T}} = \left(\sqrt{\pi(1)}, \ldots, \sqrt{\pi(n)} \right)^{\mathsf{T}}$$

holds.

A sketch of proof (4/4)

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right|$$

$$= \left|\sum_{t=0}^{T-1}\sum_{v\in V}\sum_{u\in N(v)}\left(Z_{v,u}^{(t)} - \chi_v^{(t)}P(v,u)\right)P^{T-t-1}(u,w)\right|$$

by Lemma 1

by "reversible"

$$\leq \sum_{t=0}^{T-1}\sum_{v\in V}\sum_{u\in N(v)}\left|\left(Z_{v,u}^{(t)} - \chi_v^{(t)}P(v,u)\right)\sqrt{\frac{\pi(w)}{\pi(v)}}\sum_{j=1}^{n}(\lambda_j)^{T-t-1}b_j(v)b_j(u)\right|$$

$$\leq \frac{\sqrt{\pi(w)}}{\sqrt{\pi_{\min}}}\sum_{t=0}^{T-1}|\lambda_2|^{T-t-1}\sum_{v\in V}\sum_{u\in N(v)}\sum_{j=2}^{n}\left|\left(Z_{v,u}^{(t)} - \chi_v^{(t)}P(v,u)\right)\right||b_j(v)b_j(u)|$$

$$\leq \frac{\sqrt{\pi(w)}}{\sqrt{\pi_{\min}}}\cdot\frac{1}{1-\lambda_2}\sum_{v\in V}\sum_{u\in N(v)}\sum_{j=2}^{n}\left|\left(Z_{v,u}^{(t)} - \chi_v^{(t)}P(v,u)\right)\right|$$

note: λ_1=1

$$\leq \frac{\sqrt{\pi(w)}}{\sqrt{\pi_{\min}}}\cdot\frac{m(n-1)}{1-\lambda_2}\cdot 2(\lg M + 1)$$

Main Thm.

Thm. [Shiraga, Yamauchi, K., Yamashita 12+]

If P is ergodic and reversible, then

for any ini. config., for any $w \in V$, for any time t,

$$\left|\chi_w^{(T)} - \mu_w^{(T)}\right| < \sqrt{\frac{\pi_w}{\pi_{\min}}} \cdot \frac{m(n-1)}{1-\lambda^*} \cdot 2(\lg M + 1)$$

holds where, n=|V|, m=|E|, M=#tokens (=$|\chi^{(0)}|$)

λ^* is the second largest eigenvalue of P, and

π is the stationary distribution.

P is reversible if the "detailed balance equation"

$$\pi(u)\, P(u,v) = \pi(v)\, P(v,u)$$

holds for any $u,v \in V$.

often assumed
in the context of MCMC

Remark

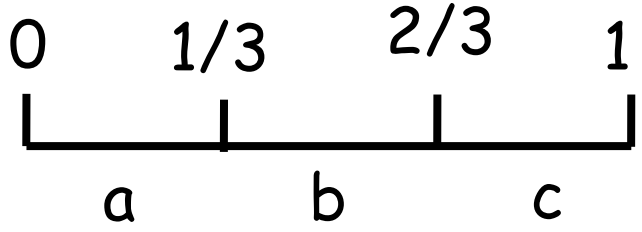
Based on the idea of "rejection sampling,"

(a version of) functional-router model emulates

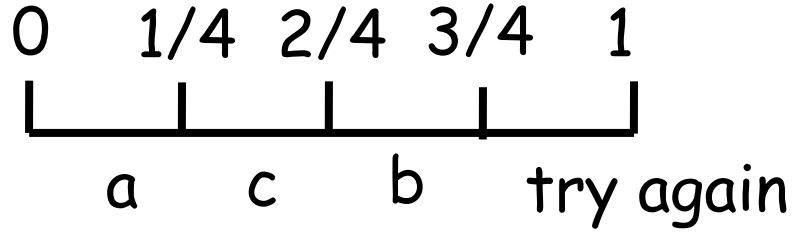
the rotor-router model (w/ rational trans. prob.)

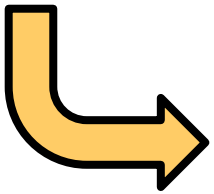
rotor-router: $\rho=\langle a,b,c\rangle$

(naive) functional-router



(modified) functional-router





"functional-router" is

an extension of "rotor-router"

4. Concluding remarks

Related topics

- ✓ IDLA (Internal Diffusion-Limited Aggregation)
  - Levine & Peres 2005
- ✓ Information Spreading
  - Doerr, Friedrich, & Sauerwald 2008
  - Doerr, Friedrich, Kunnemann, & Sauerwald 2009
- ✓ Hitting time, Cover time
  - Friedrich & Sauerwald 2010
  - Holroyd & Propp 2010+

Thm. [Holroyd & Propp 2010+]
single walk ver. det RW.
F^t_v := #visits (of token) between time 0 to t.
for any finite graph,
$|F^t_v/t - \pi^*_v| \leq O(mn/t)$
where π^* = the stat. ditr. of RW.

cf. [K, Koga, Makino 10+]
$|\chi^t_v/N - \pi^t_v| \leq O(mn/N)$

Future works

- ✓ fill the gap between $O((n^3/1-\lambda)\log M)$ & $\Omega(\log M)$.
- ✓ poly log(|V|) bound for comb. graphs.
- ✓ single vs multiple (blanket time vs mixing time).
- ✓ derandomization of MCMC.
- ✓ What is random?
  - ◆ pseudo random number
  - ◆ quasi Monte Carlo
  - ◆ Chaos time series

P. Gopalan, A. Klivans. R. Meka, D. Stefankovic, S. Vempala, E. Vigoda

An FPTAS for #Knapsack and Related Counting Problems, (FOCS 2011)

4. Concluding Remark.

1. Impact of randomness: ex. "finding frequent items in a stream"
2. Advanced prob. algo.: Markov chain Monte Carlo for sampling comb. obj.
3. Derandomization: "deterministic random walk

Question

"What is the property that

a randomized algorithm *really* requires for randomness?"

Current & Future works

"What is the property that

a randomized algorithm *really* requires for randomness?"

What I am doing

developing probabilistic algorithms & derandomizing

- Fastest RW (hitting time, cover time)

[Nonaka, Ono, K, Yamashita: CATS11]

- Online linear optimization on permutations

[Yasutake, Hatano, K., Takimoto, Takeda: ISAAC11]

- Can a robot-vacuum clean corners well?

[Hirahara, K, M. Yamashita: 2011]

etc.

Roomba ©iRobot

The end

Thank you for the attention.